

AD-A080 126

HOUSTON UNIV TX DEPT OF ELECTRICAL ENGINEERING

F/G 17/2

REMOTE LINK UNIT FUNCTIONAL DESIGN: AN ADVANCED REMOTE TERMINAL--ETC(U)

OCT 79 C J TAVORA, J R GLOVER, G W BATTEN

F33615-78-C-1634

AFAL -TR-79-1176

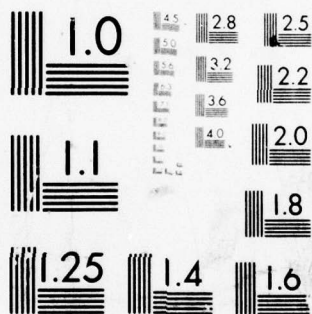
NL

UNCLASSIFIED

1 OF 3

AD
A080126





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A 080126

AFAL-TR-79-1176

LEVEL



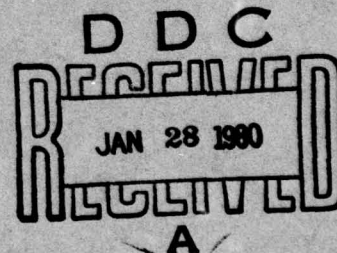
REMOTE LINK UNIT FUNCTIONAL DESIGN
An Advanced Remote Terminal For MIL-STD 1553B

UNIVERSITY OF HOUSTON
ELECTRICAL ENGINEERING DEPARTMENT
HOUSTON, TEXAS 77004

OCTOBER 1979

DDC FILE COPY

TECHNICAL REPORT AFAL-TR-79-1176
Interim Report for Period 2 October 1978 — 31 August 1979



Approved for public release; distribution unlimited.

AIR FORCE AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

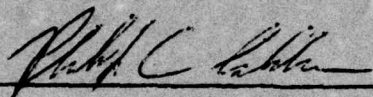
80 1 28 004

NOTICE

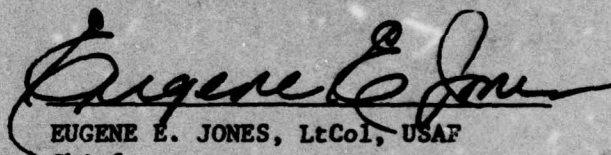
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

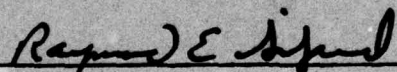


Philip C. Goldman
Project Engineer
System Design Group



EUGENE E. JONES, LtCol, USAF
Chief
Avionic Systems Engineering Branch

FOR THE COMMANDER



RAYMOND E. SIFERD, Colonel, USAF
Chief
System Avionics Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAL/AAA, W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER AFAL-TR-79-1176	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) REMOTE LINK UNIT FUNCTIONAL DESIGN: AN ADVANCED REMOTE TERMINAL FOR MIL-STD-1553B	5. TYPE OF REPORT & PERIOD COVERED Interim Report, For Subject Oct 1978-31 Aug 79		
7. AUTHOR(s) Carlos J. Tavora, John R. Glover, Jr., George W. Batten	8. CONTRACT OR GRANT NUMBER(s) F33615-78-C-1634		
9. PERFORMING ORGANIZATION NAME AND ADDRESS Electrical Engineering Department University of Houston 4800 Calhoun Blvd., Houston, TX 77004	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2003-01-19 01		
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory (AAA) Air Force Systems Command Wright-Patterson AFB, OH 45433	12. REPORT DATE October 1979		
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Co	13. NUMBER OF PAGES 219		
		15. SECURITY CLASS. (of this report) Unclassified 12/225	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.			
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)			
18. SUPPLEMENTARY NOTES			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Remote Terminal, Data Acquisition and Control, Subsystem Interface, Signal Interface, Subsystem Data Multiplexer, Subsystem Identification.			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report presents a functional design of the Remote Link Unit (RLU). The RLU is an intelligent remote terminal which is compatible with hierarchical distributed processing. It has interface modules capable of processing sub-system data on-the-fly and supporting a universal signal interface which can be configured under software control to input or output analog and digital signals of varied types. Electronic nameplates containing identification, interface requirements and signal processing programs are attached to avionics			

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

401 227

tab

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

20. subsystems to support the RLU operational features. The RLU stand-alone processing capabilities provide fault tolerance and facilitate maintenance.

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

FOREWORD

This report was prepared by the University of Houston, Houston, Texas, on Air Force Contract No. F33615-78-C-1634 entitled "The Remote Link Unit: An Advanced Remote Terminal for MIL-STD-1553A".

The work was administered under the direction of the System Avionics Division of the Air Force Avionics Laboratory. Mr. James Bain of the System Design Group was the Project Engineer. This responsibility was transferred to Mr. Robert Heuman and subsequently to Mr. Phil Goldman. Mr. John Ostgaard of the Hardware Group was the Liaison Engineer on matters of DAIS design compatibility.

This interim report summarizes the design accomplished under the contracted study. The Principal Investigator and Program Manager for this contract was Dr. Carlos J. Tavora. Drs. John R. Glover and George W. Batten were Co-Investigators. Dr. Glover was responsible for detailed design found in Sections II, III, V, and VI. Dr. Batten was responsible for detailed design in Section VII and the material in Appendices C and D.

The technical and administrative support provided by Lieutenant Colonel Eugene Jones and Mr. Robert Harris was a significant factor in the success of this study.

Accession For	
NTIS GEM&I	<input checked="checked" type="checkbox"/>
DDC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Availand/or special
A	

TABLE OF CONTENTS

SECTION	PAGE
I INTRODUCTION	1
1.1 THE RLU CONCEPT	1
1.2 FROM RT TO RLU	2
1.3 DESIGN OBJECTIVES	5
1.4 REPORT ORGANIZATION	6
II DAIS/RLU INTERFACE	7
2.1 COMMUNICATION ON THE MULTIPLEX BUS	7
2.1.1 RLU and Subsystem Addressing	7
2.1.2 RLU Address Decoding	8
2.1.3 Synchronous and Asynchronous Messages	8
2.2 SYSTEM INITIALIZATION AND ADDRESS ASSIGNMENT	13
2.2.1 System Tables	13
2.2.2 Initialization Procedure	15
2.3 ERROR/FAILURE MANAGEMENT	18
2.3.1 Retry Procedures	18
2.3.2 Reconfiguration	18
2.4 IMPACT ON DAIS	18
III LINK MODULE	20
3.1 LINK MANAGER INTERFACE	20
3.1.1 Shared Memory Organization	20
3.1.2 Data Transfer Operations	26
3.1.3 Data Transfer Handshake	30

TABLE OF CONTENTS (CON'T.)

SECTION	PAGE
3.2 LINK MODULE INTERNAL ARCHITECTURE	33
3.2.1 Organization	33
3.2.2 Initialization and Configuration	37
3.2.3 Output Data Transfers	40
3.2.4 Input Data Transfers	42
3.2.5 Error Processing	43
3.3 LINK MODULE TESTING	45
IV INTERFACE CONFIGURATION ADAPTER	46
4.1 ARCHITECTURE	46
4.2 SIGNAL AND BUFFER DATA INTERFACES	50
4.2.1 Serial Signals	59
4.3 OPERATIONAL CONTROL AND TIMING	61
4.3.1 ADC Operation	63
4.3.2 Serial Output Operation	64
4.3.3 Serial Input Operation - Refresh Mode	67
4.3.4 Serial Input Operations - Flag Mode	67
4.4 TESTING	69
4.4.1 Signal Interface Testing	69
4.4.2 ADC Testing	71
4.4.3 Serial I/O Testing	71
V SUBSYSTEM INFORMATION CHANNEL	73
5.1 COMMUNICATION PROTOCOL	77
5.2 COMMANDS AND STATUS	77
5.3 ELECTRONIC NAMEPLATE ARCHITECTURE	81

TABLE OF CONTENTS (CONT'D.)

SECTION	PAGE
5.4 TESTING PROCEDURE	86
VI LINK MANAGER	87
6.1 LINK MANAGER ARCHITECTURE	88
6.1.1 Multiplex Terminal Unit	88
6.1.2 Shared Memory Interface	91
6.1.3 Mapping EAROM	91
6.1.4 Link Control Unit	93
6.1.5 MTU Control Interface	96
6.1.6 Maintenance Port	96
6.1.7 Manager Processor	97
6.2 LCU ARCHITECTURE	97
6.2.1 Microinstruction Format	97
6.2.2 LCU Functional Components	99
6.2.3 Accessing the Mapping EAROM	101
6.2.4 Data Transfer Operations	103
6.2.5 Mode Code Operations	106
6.2.6 SA 1 Operations	107
6.2.7 Status Word	107
6.3 SOFTWARE ORGANIZATION	108
6.3.1 SA 1 Processing - LMG Commands	112
6.3.2 Initialization and Configuration	114
6.3.3 Error Analysis and Failure Management	118
6.3.4 Mode Code Processing	118
6.3.5 Asynchronous Service Requests	119

TABLE OF CONTENTS (CON'T.)

SECTION	PAGE
6.3.6 Local Processing	121
6.3.7 Maintenance Support	121
VII SUBSYSTEM SOFTWARE	123
7.1 DATA I/O TASK OPERATION	123
7.2 SUBSYSTEM DIAGNOSTIC TASK OPERATION	126
7.3 DATA I/O TASKS: NAMEPLATE ENTRY ORGANIZATION	126
7.4 CONFIGURATION TASK OPERATION	129
7.5 THE DEVICE CONTROL TABLES ICT AND OCT	129
7.6 EXEC FACILITIES FOR THE USER	133
7.7 SUMMARY OF REQUIRED USER-WRITTEN SOFTWARE	133
7.8 A SUBSYSTEM EXAMPLE	134
7.8.1 The Position Sensing Subsystem	135
7.8.2 Signal Handling	137
7.8.3 Software Tasks	140
VIII IMPACT ON DAIS	152
8.1 CHANGES IN PALEFAC	152
8.2 CHANGES IN THE MASTER EXECUTIVE	154
8.3 CHANGES IN THE SUBSYSTEMS	155
APPENDIX A - THE REMOTE LINK UNIT-AN ADVANCED REMOTE TERMINAL CONCEPT.	157
APPENDIX B - PRESENT DAIS REMOTE TERMINALS	163
APPENDIX C - SYSTEM SOFTWARE	183
APPENDIX D - PROCESSING SYNCHRO SIGNALS	205

LIST OF ILLUSTRATIONS

FIGURE		PAGE
1	Remote Terminal (RT) Diagram	3
2	Remote Link Unit (RLU) Diagram	4
3	Flow of Commands Through RT's and RLU's	9
4	RLU Address Decoding	10
5	Asynchronous Message Operation (RLU)	12
6	Terminal Configuration Table (TCT)	14
7	Device Configuration Table (DCT)	14
8	Initial Polling Loop for RT's and RLU's	16
9	Architecture of Link Module (LM)	21
10	LM's 128-word Shared Memory Interface	23
11	Mapping of Addresses in the LM	27
12	Command/Data Flow for Output Data Transfer	28
13	Data Transfer Handshake Protocol	31
14	Initialization Sequence	38
15	Data I/O Task Execution	41
16	LM Error Processing	44
17	Internal Organization of the Interface Configuration Adaptor (ICA)	49
18	Address Allocation for ICA Interface Buffers	52
19	Organization of the Data Input (DATAIN) Section of the Inter- face Buffer	53
20	Organization of the Data Output (DATAOUT) Section of the Inter- face Buffer	54
21	Organization of Parameter Section of the Interface Buffer . .	55

LIST OF ILLUSTRATIONS (CONT'D.)

FIGURE		PAGE
22	ICA Signal Interface	57
23	Handshake Between LM and Subsystem for Serial Data Transmission	60
24	Operating States of Each ICA Group and Permissible Commands at Each State	62
25	Timing Diagram for Conversion of Analog Inputs	65
26	Serial Output Operation	66
27a	Serial Input Operation - Refresh Mode	68
27b	Serial Input Operation - Flag Mode	68
28	Communication with Nameplate through the Subsystem Information Channel	75
29	Master Nameplate Configuration	76
30	Message Format on the Subsystem Information Channel (SIC) . .	78
31	Nameplate Architecture	82
32	State Diagram and Timing Diagram	84
33	Nameplate Memory Organization	85
34	Architecture of Link Manager	89
35	LMG Memory and I/O Address Space	90
36	Address Word Format for Shared Memory	92
37	Mapping EAROM Organization	94
38	Example of Mapping EAROM Entries	95
39	Format of LCU Microinstruction	98
40	Basic Architecture of the LCU	100
41	Addresses Used by LCU on LMG Bus	102
42	Message Transfer Handshake Protocol	104
43	LMG Initialization Sequence	115

LIST OF ILLUSTRATIONS (CONT'D.)

FIGURE		PAGE
44	Asynchronous Message Operation (LMG and LM)	120
45	Data I/O Task	124
46	Typical Organization of SDM	127
47	Nameplate Entries for Subsystem Data I/O Tasks	128
48	Configuration Task Operation	130
49	Allocation of LM Memory User Area	131
50	Position Sensing Subsystem	136
51	Switch Operation	136
52	Subsystem Signals	138
53	Data I/O Tasks	141
54	Task MEAS	146
55	Task CALIB	147
56	Task LIMIT	150
57	Subsystem Diagnostic Module	151
B-1	DAIS Message Word Format	165
B-2	DAIS Message Format	165
B-3	RT Command Decoding	166
B-4	Receive Sequence	167
B-5	Transmit Sequence	167
B-6	Interaction Between RT Registers	169
B-7	Synchronous Message Tables (Master)	171
B-8	Synchronous Message Tables (Remote)	172
B-9	Asynchronous Message Tables (Master)	174
B-10	Asynchronous Message Tables (Remote)	176
C-1	LM System Software	185

LIST OF ILLUSTRATIONS (CONT'D.)

FIGURE	PAGE
C-2 Device-Process and Controlling Programs	189
C-3 Flow of Control in a Device Handler	190
C-4 Active Task and Controlling Programs	197
C-5 Flow of Control in Programs Responsible for a Task	198
C-6 Data Sources and Sinks for Data I/O Tasks	201
C-7 Data I/O Task	202
C-8 Data I/O Task Operation	203

LIST OF TABLES

TABLE	PAGE
1 FEATURES OF REMOTE TERMINALS	3
2 FEATURES OF THE REMOTE LINK UNIT	4
3 SM COMMAND AREA	24
4 SM STATUS AREA	25
5 SYSTEM TASKS	34
6 SYSTEM TABLES	36
7 ICA INTERFACE TO SUBSYSTEMS	47
8 ICA INTERFACE TO THE LINK MODULE	51
9 NAMEPLATE COMMANDS	53
10 LMG SYSTEM TASKS	109
11 LMG SYSTEM TABLES	111
12 LMG COMMANDS	113
13 I/O CONTROL TABLES	143
14 ICA CONTROL TABLE AND BUFFER FOR THE CONFIGURATION TASK	145
B-1 DAIS SUBSYSTEM INTERFACE SIGNALS	179
C-1 FUNCTIONS OF DEVICE HANDLER COROUTINES	192

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

ACSYNC	- 400 Hz Reference Signal
A/D	- Analog-to-Digital
ADC	- Analog-to-Digital Converter
AIB	- Analog Input Bus
ARF	- Asynchronous Request Flag
ARQ	- Activity Request Queue
ATL	- Active Task List
BADPAR	- Bad Parity Flag
BAV	- Buffer Available Bit
BC	- Notation for BCIU in Flow Charts
BCIU	- Bus Control Interface Unit
BININ	- Binary Input
BINOUT	- Binary Output
BIT	- Built-In-Test
BTK	- Buffer Taken Bit
CBA	- Command Buffer Address
CBUSY	- Channel Busy
CH	- Channel
COMP	- Comparator
CONFIG	- Configure (Command)
CPU	- Central Processing Unit
CRT	- Cathode Ray Tube
CT#	- Control Table
DAC	- Digital-to-Analog Converter
DAIS	- Digital Avionics Information System
DATAIN	- Data Input
DATAOUT	- Data Output
DCP	- Data Conversion Program
DCSW	- Data Conversion Status Word
DCT	- Device Configuration Table
DDB	- Data Block Descriptor
DEMUX	- Demultiplexer
DIB	- Digital Input Bus
DMA	- Direct Memory Access
DOB	- Output Data Bus
DR	- Data Ready
EAP	- Error Analysis Program
EAR	- Error Analysis and Recovery
EAROM	- Electrically Alterable Read Only Memory
EID	- Electronic Identification Number
EOC	- End-of-Conversion

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS (CONT'D.)

EPROM	- Erasable/Programmable Read Only Memory
ERCNT	- Error Counter
EROM	- Same as EPROM
ERR	- Error Bit
ERSW	- Error Recovery Status Word
ETI	- Executive Task Initiator
ETS	- Executive Task Scheduler
FLAG/ACK	- Flag and Acknowledge Serial Data Signal
FLG	- Flag Bit
HILEVL	- High Level Voltage
HWSW	- Hardware Status Word
ICA	- Interface (Configuration, Control) Adaptor
IC	- Interface Communicator
IC.C	- Interface Communicator Continuator
IC.E	- Interface Communicator Error Processor
IC.I	- Interface Communicator Initiator
ICT	- Input Control Table
IHSW	- Input Hardware Status Word
ILPT	- Instruction List Pointer Table
IM	- Interface Modules
INAN	- Analog Input Line
INIT	- Initialize Bit
I/O	- Input/Output
IOTC	- I/O Task Controller
IOTM	- I/O Task Module
ISR	- Interrupt Service Routine
LA	- Link Address
LCU	- Link Control Unit
LDSTB	- Buffer Load Strobe
LM	- Link Module
LMD	- Link Module Directory
LMG	- Link Manager
LMP	- Link Module Processor
LOLEVL	- Low Level Voltage
LPD	- Local Program Directory
LSB	- Least Significant Bit
MAS	- Memory Address Space
MC	- Mode Code
MCI	- MTU Control Interface

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS (CONT'D.)

MCU	- Microprogram Control Unit
MDAC	- Multiplying Digital-to-Analog Converter
ME	- Notation for Master Executive in Flow Charts
MGP	- Manager Processor
MINK	- Master Instruction Keys Table
MP	- Maintenance Port
MSB	- Most Significant Bit
MTU	- Multiplex Terminal Unit
MUX	- Multiplexer

NCOM	- Number of Words Allocated to LM Common
NCW	- Nameplate Control Word
NIC	- Nameplate Interface Controller
NP	- Electronic Nameplate
NPDIAG	- Run Nameplate Diagnostic (Command)
NPIM	- NP Initialization Module
NPINIT	- Run Nameplate Initialization (Command)
NPT	- Nameplate Topology Table
NSE	- Number of SCT Entries

OCT	- Output Control Table
OHSW	- Output Hardware Status Word
OUTAN	- Analog Output Line
OUTDIS	-

PBD	- Pointer Block Descriptor
PDCP	-
PGI	- Palefac Global Input
PM	- Process Manager
PM.C	- Continuator
PM.I	- PM into a Initiator
PM.R	- Resource Allocator
PSDM	- Pointer to SDM
PRGMLD	- Program Load (Command)
PTR	- Pointer

RAM	- Random Access Memory
RAT	- Remote Asynchronous Table
RCT	- RLU Configuration Table
RDY	- Ready Bit
REL	- Release Bit
REQ	- Request
REQ/LOK	- Request and Lockout Serial Data Signal
RFRSH	- Refresh
RL	- Notation for RLU in Flow Charts

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS (CONT'D.)

RLU	- Remote Link Unit
ROM	- Read Only Memory
RT	- Remote Terminal
SA	- Subaddress
SCAN	- Type of Scan Used in ICA
SCT	- Subsystem Configuration Table
SDC	- Subsystem Data Channel
SDM	- Subsystem Diagnostic Module
SDP	- Subsystem Diagnostic Program
SDT	- Subsystem Diagnostic Task
SERCLK	- Clock Line
SERDAT	- Data Line
S/H	- Sample-and-Hold Circuit
SIC	- Subsystem Information Channel
SIL	- Synchronous Instruction List
SM	- Shared Memory
SMI	- Shared Memory Interface
SNAKE	- Subaddress Name Keys Table
SPD	- Subsystem Program Directory
SS	- Subsystem
SSDIAG	- Subsystem Diagnostic (Command)
STATUS	- Status (Command)
STATW	- Status Word
STB	- Strobe
STD	- System Task Directory
SYNPTR	- Synchronous Pointer Table
TA	- Terminal Address
TC	- Traffic Controller
TC.C	- TC into a Completion Evaluator
TC.I	- Interrupt Service Routine
TCU	- Timing and Control Unit
TCT	- Terminal Configuration Table
TIB	- Task Input Buffer
TIM	- Time Controller
TOAD	- Transmit Originate Address Table
TOB	- Task Output Buffer
T/R	- Transmit/Receive
TRSHLD	- Threshold Level
UART	- Universal Asynchronous Receiver/Transmitter
FIFO	- First In-First Out
WC	- Word Count
WSC	- Word Subcount
WRDGNT	- Word Count (ICA)
XFRTBL	- Transfer Table (Command)

SECTION I

INTRODUCTION

This document describes the functional design of the Remote Link Unit (RLU) which constitutes a new architectural approach to the implementation of remote terminals. This design demonstrates the feasibility of incorporating the RLU concept as a part of DAIS. This report addresses three major areas of concern: RLU internal structure (i.e., hardware architecture and software organization), the RLU interface to DAIS (protocol with master and local executives) and the RLU interface to DAIS subsystems (interface configuration and software support). This report was prepared under contract #F33615-78-C-1634.

1.1 THE RLU CONCEPT

The remote link unit (RLU) is an evolution of the remote terminal. It provides a complete and direct interface between a CPU and remote subsystems. The RLU has universal interface modules (referred to as Link Modules) which are able to identify interfaced subsystems and to configure data and timing signals to match the subsystem requirements. The subsystem identification and interface requirements are provided by an electronic nameplate which is interrogated by the link module. An additional feature of the electronic nameplate is that it will store programs for subsystem handling, engineering unit conversion, and subsystem calibration. These programs, when uploaded to the link module, will make the subsystem peculiarities invisible to the DAIS processors. Subsystems interfaced through RLUs may be relocated or substituted without requiring changes in the CPU software to correct device ad-

dresses or conversion constants. The electronic nameplate will also simplify inventory control, automatic calibration and maintenance of subsystems. A complete description of the major features of the RLU is outlined in the statement of work of contract number F33615-78-C-1634 entitled, "The Remote Link Unit: An Advanced Remote Terminal for MIL-STD-1553A". An expanded description of the RLU concept is presented in Appendix A.

1.2 FROM RT TO RLU

The RLU can be viewed as an evolution of existing RT. This evolution results from the distribution of RT functions among specialized processors in the RLU. Most functions implemented in present RT's are performed by a bit slice processor referred to as the TCU. In the RLU these functions are performed by two types of processors: the link module (LM) which is an intelligent universal interface module, and the link manager (LMG) which coordinates the operation of all link modules and supports communication through the DAIS multiplex bus. By implementing the LM and LMG functions with specialized processors it is possible to achieve with the RLU highly efficient, reliable and flexible operation. A comparison between the internal organization of remote terminals and remote link units is highlighted by the diagrams presented in Figures 1 and 2 respectively. The principal features of RTs and RLUs are summarized in Tables 1 and 2.

The RLU architecture provides a well defined boundary (LMG/LM) that separates DAIS oriented from subsystem oriented processing. The

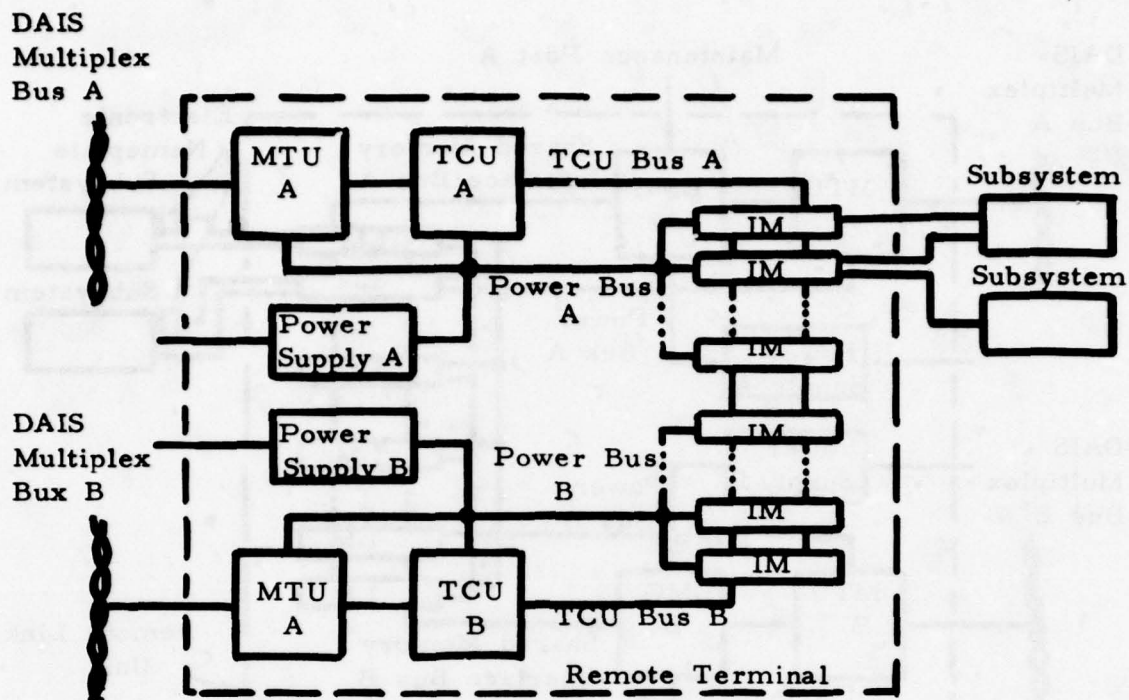


Figure 1 Remote Terminal (RT) Diagram

TABLE 1

FEATURES OF REMOTE TERMINALS

TCU Functions

- Multiplex bus monitoring and control
- Command decoding
- RT timing and control
- Self-test and interface module test
- Analog-to-digital signal conversion
- Execution of mode codes
- Routing of data traffic to and from the multiplex bus

IM Functions

- Implement a signal type interface (17 module types)
- Support self-test with redundant data path

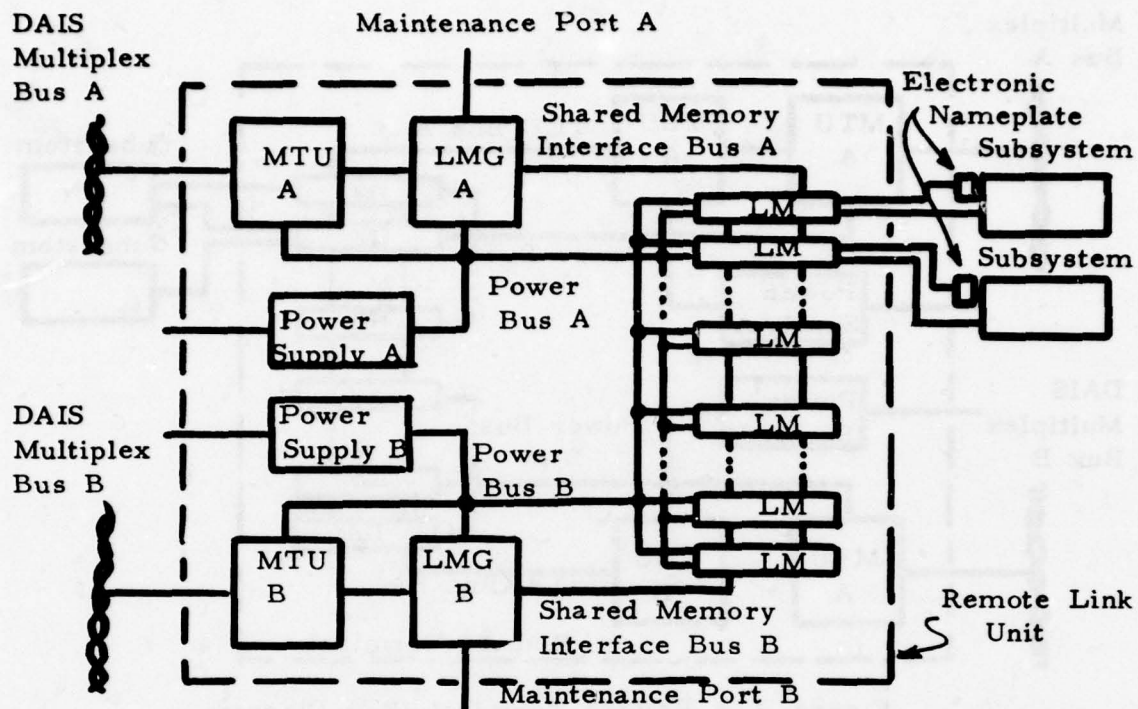


Figure 2 Remote Link Unit (RLU) Diagram

TABLE 2

FEATURES OF THE REMOTE LINK UNIT

LMG Functions

- Multiplex bus monitoring and control
- Routing of data to and from the multiplex bus
- Processing mode codes
- Supervision of local processing
- Management of subsystem interface configuration
- Management of back-up link modules and subsystems
- Support of maintenance port

LM Functions

- Configuration and control of interface to subsystem
- Conversion and formatting of data
- Self-test and subsystem test
- Control and supervision of electronic nameplates
- Concurrent operation of 4 distinct signal interfaces

LM provides subsystem designers with software control of the hardware interface configuration and a processing environment in which data conversion, error recovery and maintenance diagnostic programs may be executed. LMs are subsystem oriented programmable interface processors and may be viewed as extensions of the corresponding subsystems into an RLU.

The link manager on the other hand is designed to support the transfer of messages through the multiplex bus and to implement DAIS system functions. Its design provides the flexibility which DAIS integrators may need to provide the system with fault tolerance, quick reconfiguration capability and support to automated maintenance.

1.3 DESIGN OBJECTIVES

The functional design presented in this document was prepared to accomplish the following objectives:

1. Satisfy MIL-STD-1553B bus message structure.
2. Minimize the changes in DAIS required for its integration.
3. Be compatible with the operation of both RTs and RLUs.
4. Utilize a single type of Link Module to implement all DAIS subsystem interfaces.
5. Allow the Link Module to support both signal oriented as well as function oriented interfaces.
6. Utilize electronic nameplates to identify interfaced subsystems and to select the required interface configuration.

7. Extract from the electronic nameplate the data conversion and diagnostic programs required for operating and maintaining each subsystem.

8. Store in the electronic nameplate information regarding subsystem operational malfunction and repair records.

1.4 REPORT ORGANIZATION

This report has been organized in a manner that simplifies the design presentation and facilitates the understanding of RLU operational concepts. Section 2 describes the interaction between an RLU and the DAIS system processors. It discusses the message structure and the manner by which subsystems, link modules and the link manager are accessed. Section 3 describes the internal structure and the operation of a link module. Section 4 describes the interface configuration adapter which is the key element in providing a universal interface capability to the link module. Section 5 describes the subsystem information channel which consists of the link module's nameplate interface controller and the subsystem's nameplate. Section 6 describes the internal structure and operation of the link manager. Section 7 summarizes the requirements which must be met by and the resources available to a subsystem designer utilizing the link module interface. Section 8 evaluates the impact on DAIS resulting from integration of the RLU.

SECTION II

DAIS/RLU INTERFACE

The serial-bus communication structure of the proposed DAIS/RLU interface conforms to MIL-STD-1553B. Subsystems are addressed using the standard Terminal Address/Subaddress (TA/SA) addressing scheme. Thus, a subsystem using the proposed scheme can be configured using a mix of existing remote terminals (RT's) and the proposed remote link units (RLU's).

2.1 COMMUNICATION ON THE MULTIPLEX BUS

2.1.1 RLU and Subsystem Addressing

A Terminal Address (TA) will be assigned to each RLU in advance, as is done for RTs in the present DAIS system. This address will be hardwired into each unit using an address plug and it will be entered as an input to PALEFAC to define the system configuration.

In the present DAIS system, the mapping between subaddresses and subsystems, which is performed through an EROM in the RT, must be provided to PALEFAC by the programmer. In the proposed design, PALEFAC assigns sequential subaddresses to each RLU as needed. During system startup and initialization, each RLU is told what SA's have been assigned to its various subsystems. The SA/subsystem mapping is then stored in an electrically alterable read only memory (EAROM) in the Link Manager instead of the semi-permanent EROM used on the present RT's.

The subaddresses in an RLU may be used to provide a logical grouping of subsystems as is done in an RT, or they may be linked to an Electronic Identification Number (EID) to provide configuration and re-configuration flexibility.

The subaddresses 0 and 31 associated with any RLU are reserved for mode commands. In addition, subaddress 1 is reserved for messages to/from the Link Manager (LMG) of the RLU, rather than a subsystem. Mode commands directed to the RLU are received by the LMG for execution of the necessary operations. Subaddress 1 provides an additional means of communicating with the LMG for purposes such as obtaining the status of the RLU or any of its Link Modules (LM), for system startup initialization (see Section 2.2), or for error processing (see Section 2.3). Figure 3 illustrates the flow of commands in RTs and RLUs as a function of selected subaddresses. Since subaddresses 0, 1, and 31 are reserved, subaddresses 2-30 are available for subsystems.

2.1.2 RLU Address Decoding

When an RLU receives a command word, the TA field is compared with the RLU's terminal address encoded in the address plug. If there is a match, the subaddress field is inspected. SA's 0 and 31 will be treated as mode commands, and SA 1 indicates a message for the LMG.

For any other subaddresses, the SA field points to the location in an EAROM which maps the SA to the correct subsystem. This mapping is alterable to allow for reconfiguration in the event of a subsystem failure. Figure 4 illustrates the decoding method.

2.1.3 Synchronous and Asynchronous Messages

Since the DAIS TA/SA addressing scheme is retained in this design, all synchronous and asynchronous messages to/from RLU's are treated the same as those to/from RT's. No modifications to the present DAIS

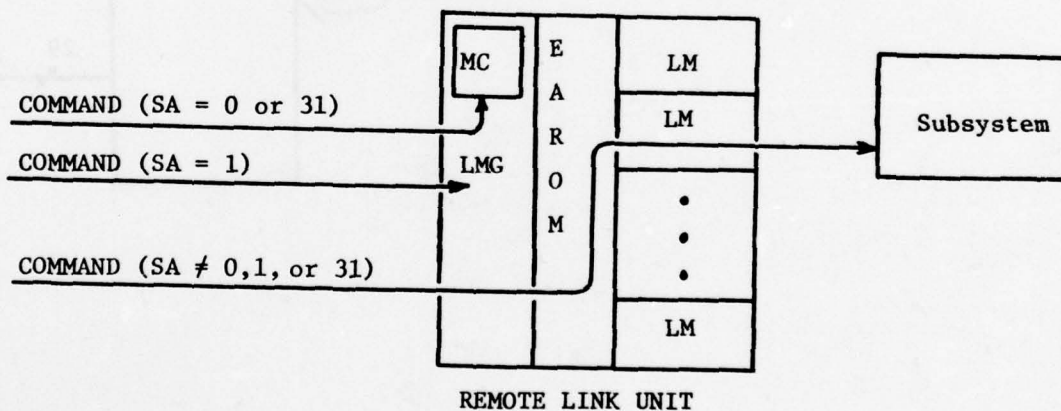
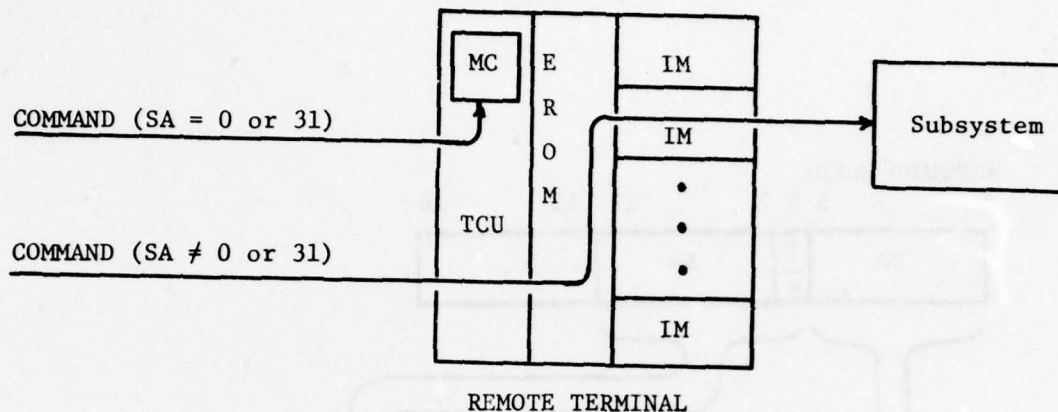


Figure 3 Flow of Commands Through RT's and RLU's

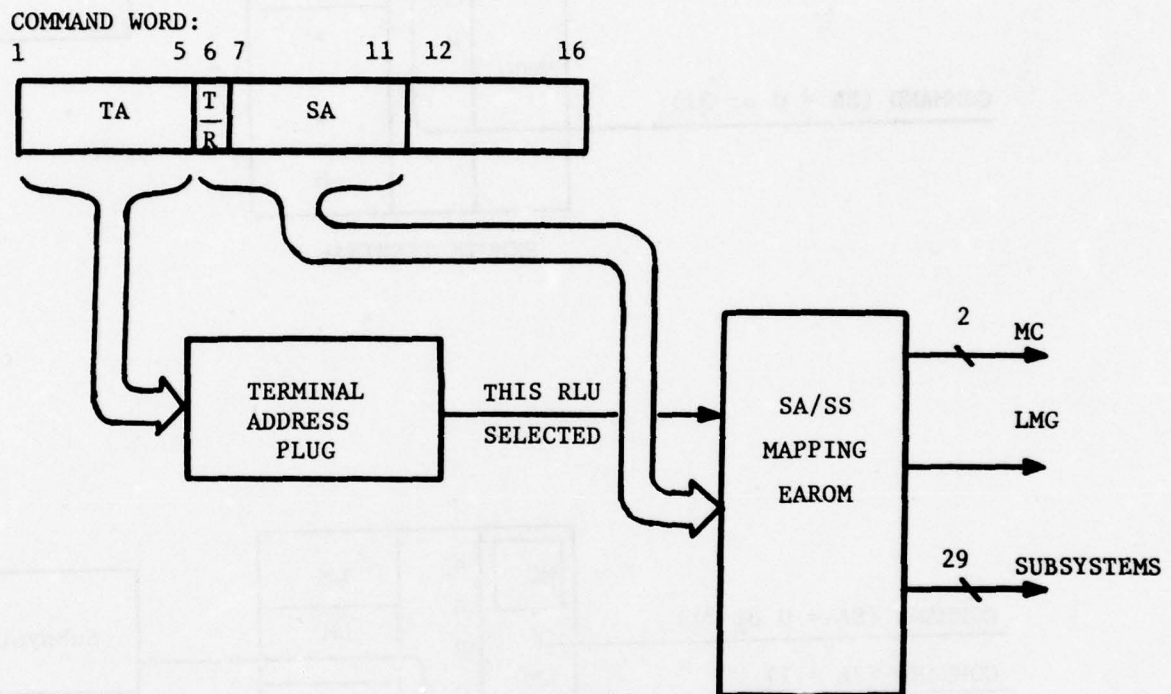


Figure 4 RLU Address Decoding

message tables are needed in either the master or local executives.

Once PALEFAC has chosen the subaddresses necessary, every subsystem then has a fixed TA/SA which is used in the message tables. There is no distinction between TA's of RTs and RLUs in these message tables. For some types of error processing a distinction is made, however, as is described in Section 2.3.

The composition of the activity register of an RLU is different from that of RTs. The physical mapping of a bit in the activity register to a corresponding channel in an interface module of an RT is not applicable to RLU's. Therefore, the activity register contents needed by the master executive for finding the correct BCIU instructions in the MINK table may be chosen in a sequential manner by PALEFAC. During startup initialization, each link module supporting an asynchronous interface to a subsystem is provided with a request vector which should be returned as the content of the activity register in reply to mode command 16 from the master executive. Figure 5 diagrams the procedure at the RLU.

When a subsystem makes a request, the Link Manager places the activity register value assigned to the subsystem in the Activity Request Queue (ARQ). The RLU alerts the master processor of an asynchronous message request by setting the activity bit in the status word. In response to mode command 16, the RLU returns the highest priority value in the ARQ. After the message has been processed, the master executive sends mode command 17 clearing that request from the ARQ.

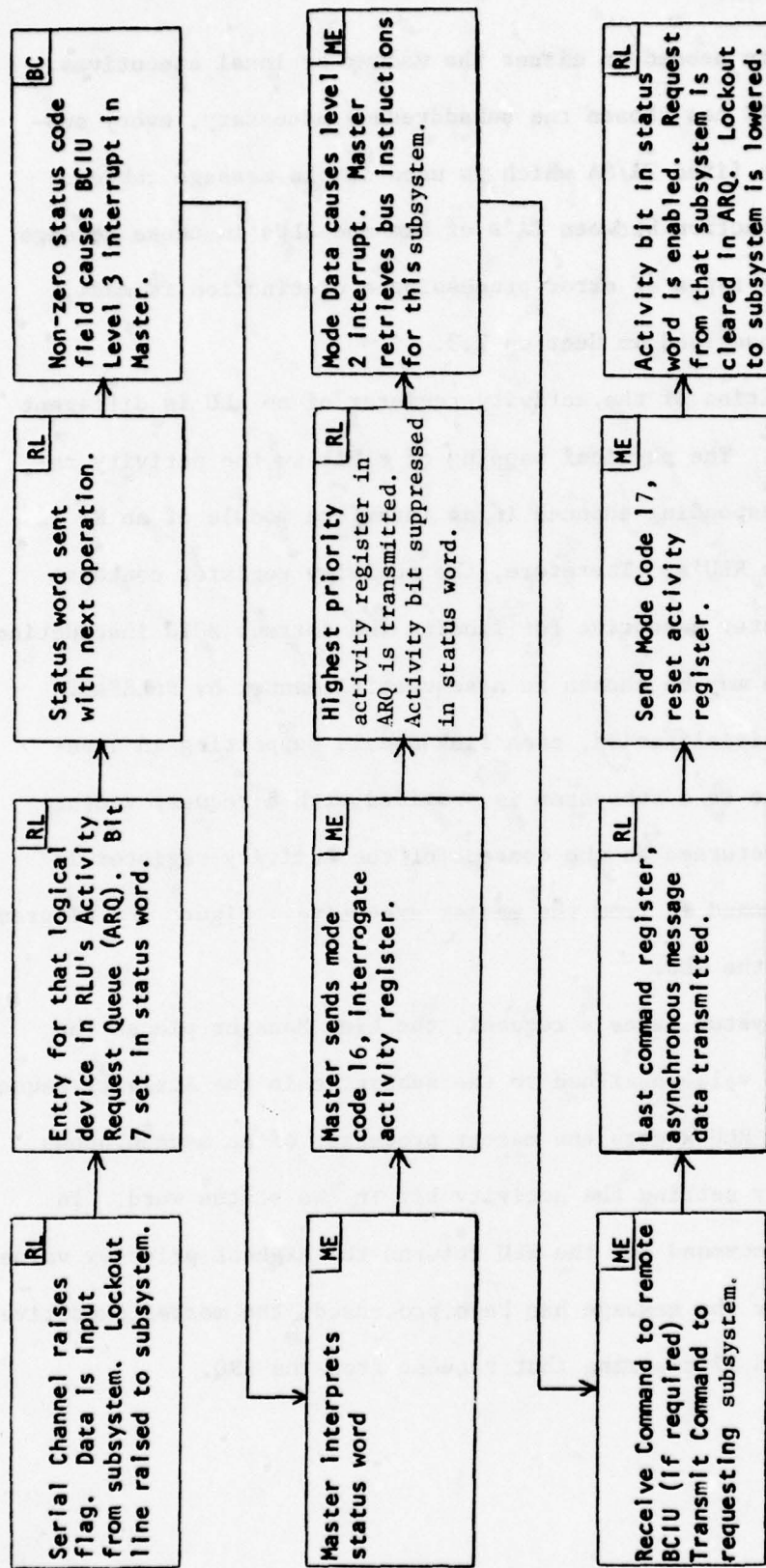


Figure 5 . Asynchronous Message Operation (RLU)

2.2 SYSTEM INITIALIZATION AND ADDRESS ASSIGNMENT

The initialization of a system containing RLU's proceeds in a manner similar to that presently used in the DAIS system. After start-up, configuration, and software verification/loading, the polling of remote terminals and remote link units takes place. The manner in which RT's and RLU's are polled constitutes the major difference between the present system initialization and that required with remote link units.

2.2.1 System Tables

At the time of system startup and initialization, the RLU's must be given information regarding the subaddresses and activity register values assigned to each subsystem by PALEFAC. In addition, two new tables are needed for this purpose, called the Terminal Configuration Table (TCT) and the Device Configuration Table (DCT).

The TCT (see Figure 6) contains the following information for each terminal address: the type of device (RT or RLU) the status of each RLU, a pointer into the DCT table, and the number of corresponding entries in DCT.

The DCT has entries grouped by RLU address (see Figure 7). Each entry has three parts: Subaddress, Electronic Identification number (EID) and interface configuration. Thus, for any RLU terminal address the DCT specifies the subsystem EID that has been assigned to each subaddress, and the interface configuration required by each subsystem. The interface configuration includes a PALEFAC-assigned contents for the activity register to be used by subsystems with asynchronous transmission capability.

Terminal Address	Required	Present	Type (T=RT, L=RLU)	RT/RLU Status	Pointer into SCT	# of SCT Entries
4	1	1	T	(Pwr sup,	-	-
5	1	0	T	MTU,	-	-
6	1	1	L	LM,	PTR6	NSE6
7	1	1	L	etc.)	PTR7	NSE7
8						
9	0	1	L		PTR9	NSE9
30						
31	Reserved for Broadcast					

Figure 6 Terminal Configuration Table (TCT)

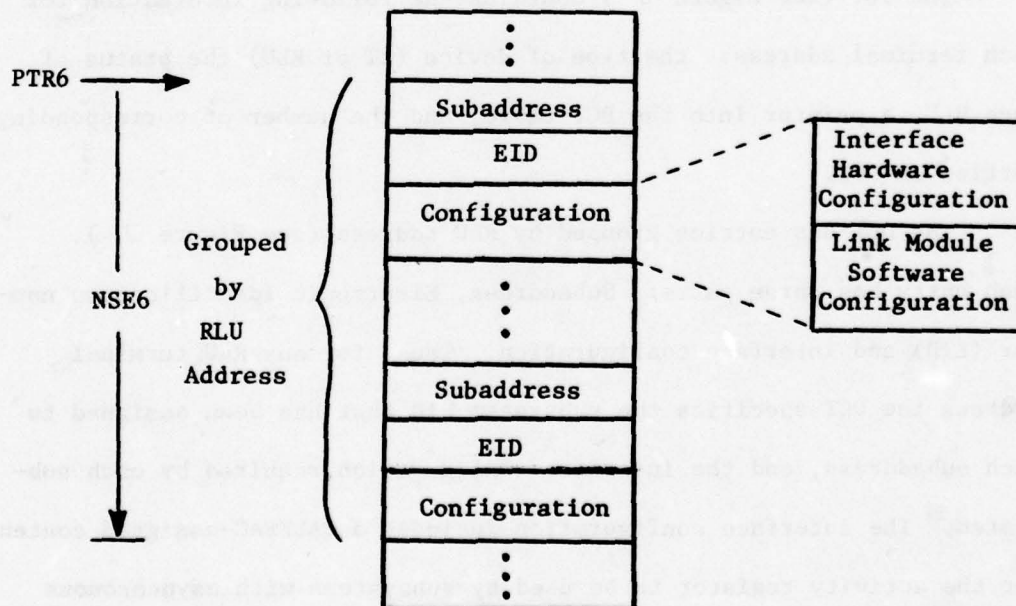


Figure 7 Device Configuration Table (DCT)

The interface configuration portion of an entry in the DCT table may contain additional information for subsystems which are connected to an RLU, but which do not have an EID or electronic nameplate. This information would include pointers to data conversion routines, subsystem diagnostic routines, and ICA configuration parameters, all to be downloaded to the RLU at initialization time.

Both the TCT and DCT are constructed by PALEFAC. However, the master executive updates status information in the TCT during initialization and during error processing.

2.2.2 Initialization Procedure

At system initialization, during the polling of RT's and RLU's, the following operations are performed by the master executive:

- Each RLU, through its TA, is polled for status. The TCT is referenced for this procedure.
- Each RLU address is informed of the subsystems which have been assigned to each subaddress, and of the activity register values to be used by asynchronous subsystems. Both TCT and DCT are used in this procedure.
- Hardware and software configuration information is downloaded to RLU's which have subsystems without an electronic nameplate.

All of these operations are performed as messages to the Link Manager of each RLU (SA=1). Figure 8 outlines the initialization sequence.

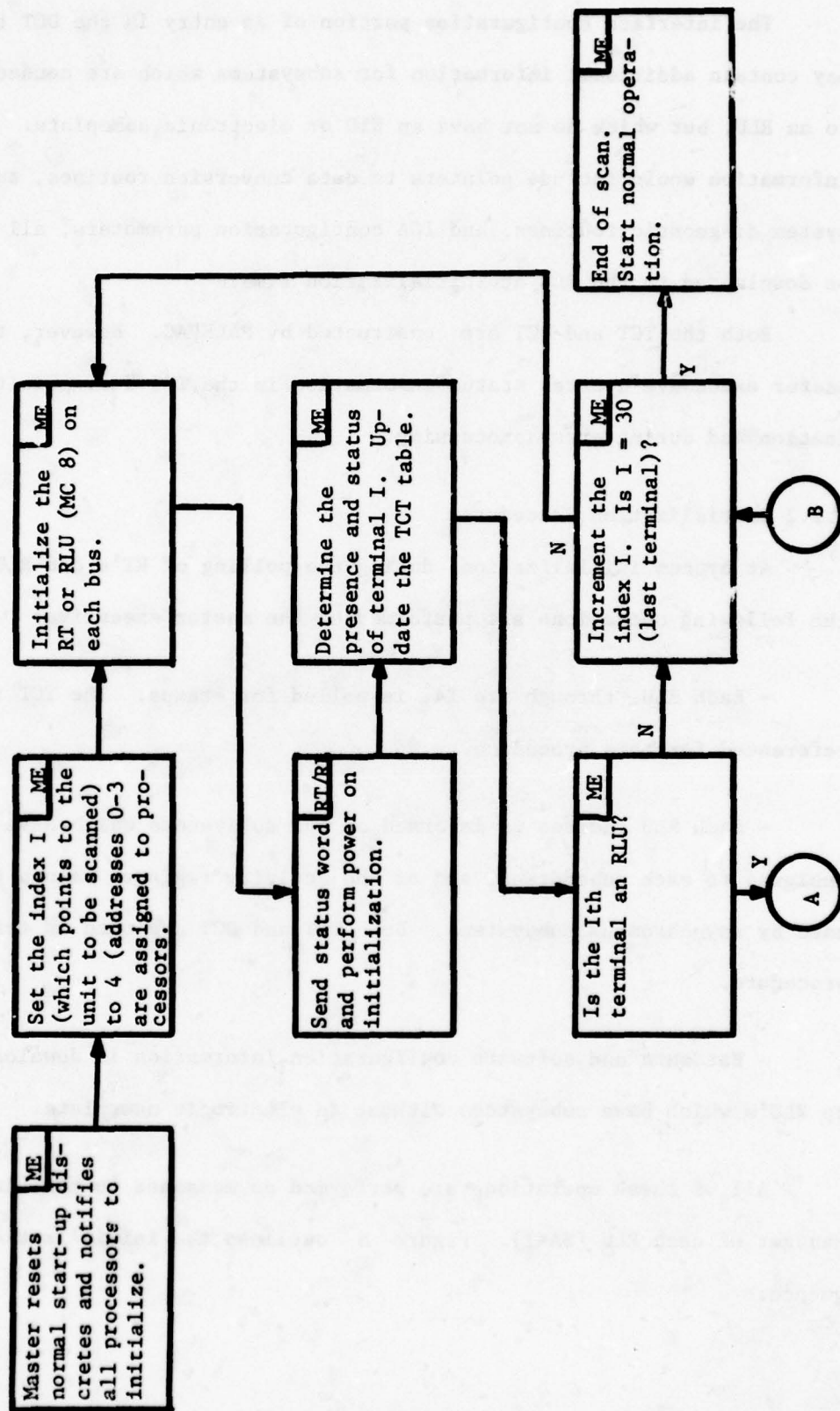


Figure 8 Initial Polling Loop for RT's and RLU's (Part 1 of 2)

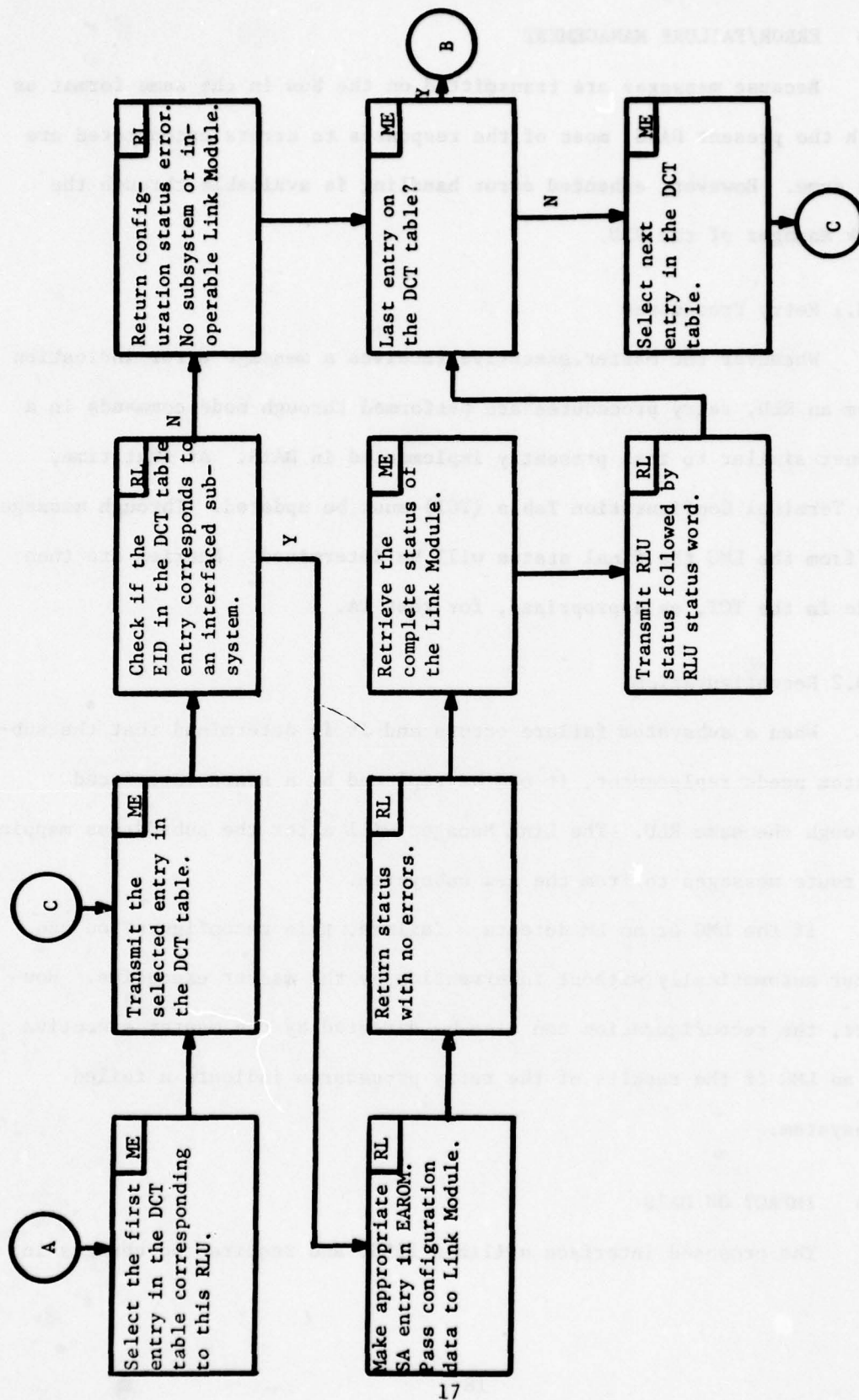


Figure 8 Initial Polling Loop for RT's and RLU's (Part 2 of 2)

2.3 ERROR/FAILURE MANAGEMENT

Because messages are transmitted on the bus in the same format as with the present DAIS, most of the responses to errors encountered are the same. However, enhanced error handling is available through the Link Manager of the RLU.

2.3.1 Retry Procedures

Whenever the master executive receives a message error indication from an RLU, retry procedures are performed through mode commands in a manner similar to that presently implemented in DAIS. At that time, the Terminal Configuration Table (TCT) must be updated. Through messages to/from the LMG the final status will be determined. Entries are then made in the TCT, as appropriate, for that TA.

2.3.2 Reconfiguration

When a subsystem failure occurs and it is determined that the subsystem needs replacement, it can be replaced by a spare interfaced through the same RLU. The Link Manager will alter the subaddress mapping to route messages to/from the new subsystem.

If the LMG or an LM detects a failure, this reconfiguration can occur automatically without intervention by the master executive. However, the reconfiguration can also be directed by the master executive to an LMG if the results of the retry procedures indicate a failed subsystem.

2.4 IMPACT ON DAIS

The proposed interface utilizes 1553B and requires no changes in

the existing DAIS hardware. The local executive software is also unaffected. Changes are required in PALEFAC and the master executive, but are not substantial. Section 8 summarizes the overall impact of the RLU on DAIS.

SECTION III

LINK MODULE

The basic architecture of the Link Module (LM) is illustrated in Figure 9 . The LM processor communicates commands, status, and data with the Link Manager (LMG) through a Shared Memory (SM) interface. The processor has its own memory, ROM, EAROM, and RAM, for storage of programs and data. The Interface Configuration Adaptor (ICA) provides a configurable universal interface to a wide range of subsystem signal types. The Nameplate Interface Controller (NIC) implements the Subsystem Information Channel (SIC) with the electronic nameplates (NP) on each subsystem so equipped. Each of these LM components is described in greater detail in the sections that follow.

The LM supervises the transfer of data between the subsystems to which it is interfaced and the LMG. Programs in the LM provide data conversion and data validity checks as the data is passed between the ICA data buffers and the shared memory interface to the LMG.

Through the ICA, the LM also directs the configuration of the universal interface to each subsystem. The required configuration information is either extracted from the subsystem's electronic nameplate via the SIC, or obtained directly from the LMG.

3.1 LINK MANAGER INTERFACE

3.1.1 Shared Memory Organization

To the Link Manager, the interface with the Link Module appears to be 128 words of Shared Memory (SM). Control and timing signals are included to provide arbitration of access to the SM by the LMG and the

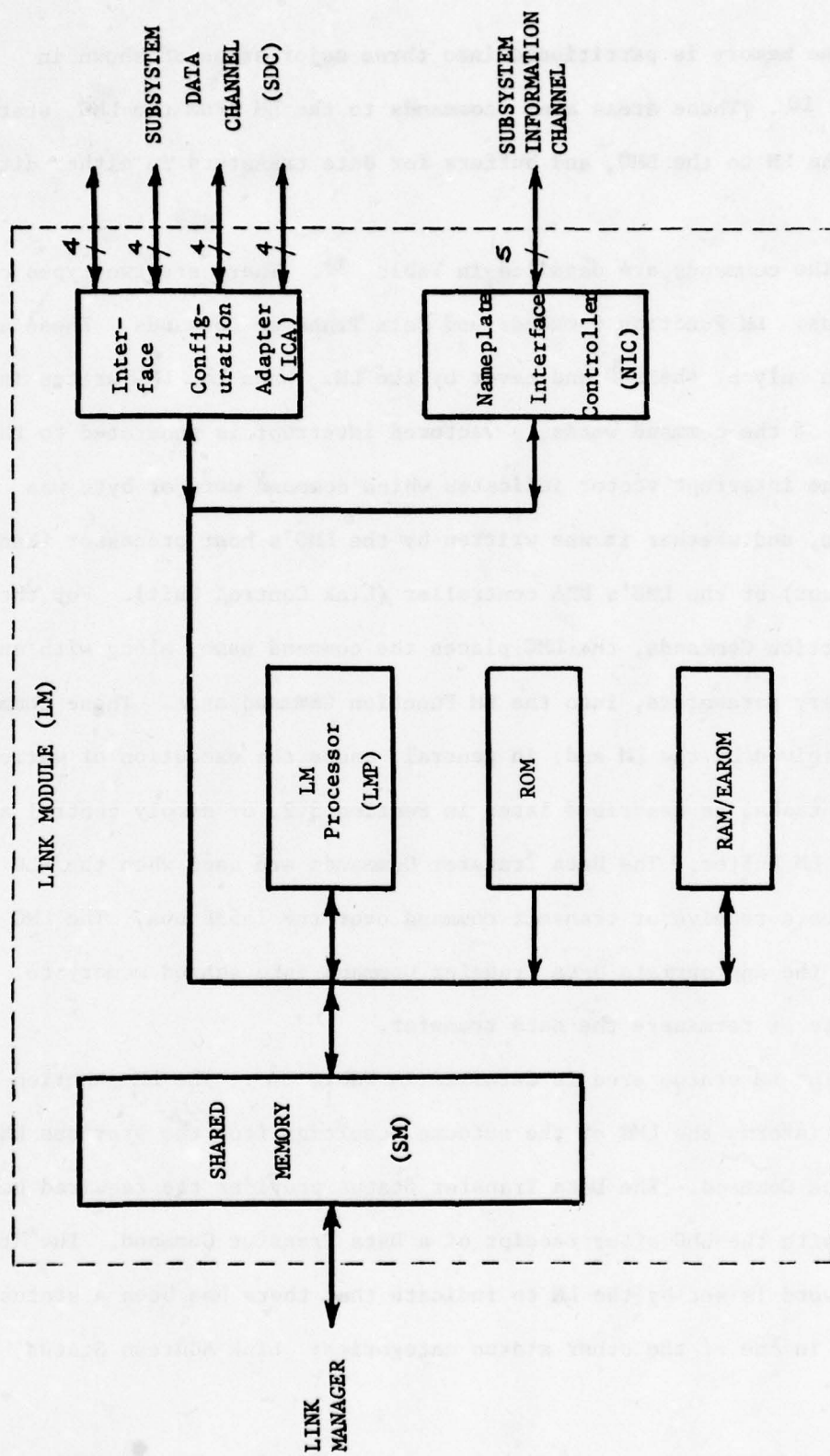


Figure 9 Architecture of Link Module (LM)

LM. The memory is partitioned into three major areas as shown in Figure 10. These areas are: commands to the LM from the LMG, status from the LM to the LMG, and buffers for data transfers in either direction.

The commands are detailed in Table 3. There are two types of commands: LM Function Commands and Data Transfer Commands. These are written only by the LMG and never by the LM. When the LMG writes into either of the command words, a vectored interrupt is generated to the LM. The interrupt vector indicates which command word or byte was written, and whether it was written by the LMG's host processor (Manager Processor) or the LMG's DMA controller (Link Control Unit). For the LM Function Commands, the LMG places the command name, along with any necessary parameters, into the LM Function Command area. These commands are received by the LM and, in general, cause the execution of various system tasks, as described later in Section 3.2, or simply control access to the LM Buffer. The Data Transfer Commands are used when the RLU receives a receive or transmit command over the 1553B bus. The LMG places the appropriate Data Transfer Command into shared memory to initiate or terminate the data transfer.

The SM status area is detailed in Table 4. The LM Function Status informs the LMG of the outcome resulting from the previous LM Function Command. The Data Transfer Status provides the required handshake with the LMG after receipt of a Data Transfer Command. The Status Alert word is set by the LM to indicate that there has been a status change in one of the other status categories: Link Address Status,

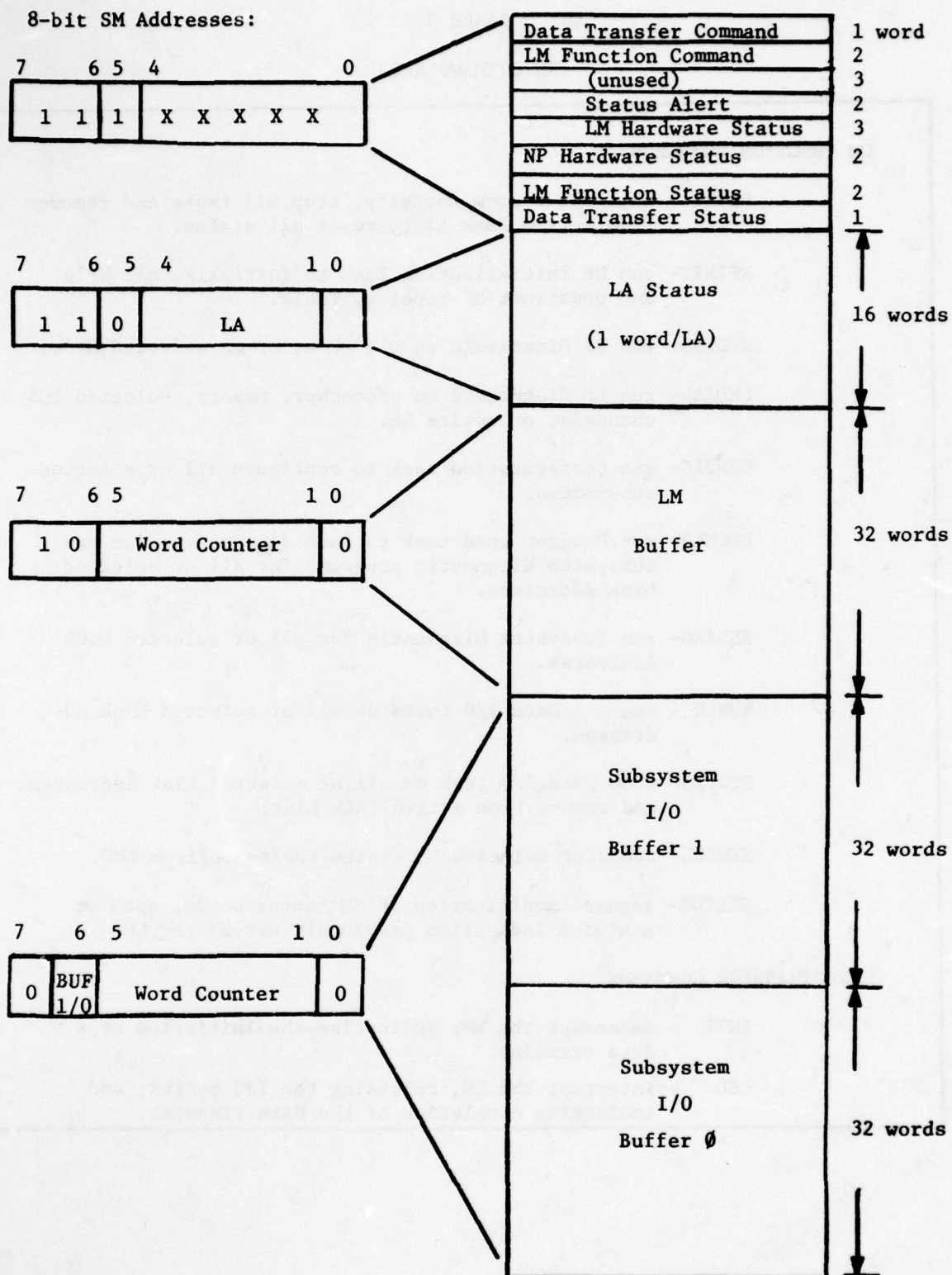


Figure 10 LM's 128-word Shared Memory Interface

TABLE 3

SM COMMAND AREA

LM FUNCTION COMMANDS

- RESET - halt all system activity, stop all tasks and remove from Active Task List, reset all status.
- NPINIT- run NP Initialization Task to initialize all NP's and construct NP Topology Table.
- NPDIAG- run NP Diagnostic on all NP's, or on selected NP's.
- LMDIAG- run LM Diagnostic on processor, memory, selected ICA channels, or entire LM.
- CONFIG- run Configuration task to configure all or selected subsystems.
- PRGMLD- run Program Load task to load data conversion and subsystem diagnostic programs for all or selected Link Addresses.
- SSDIAG- run Subsystem Diagnostic for all or selected Link Addresses.
- RUNIO - run Data I/O tasks on all or selected Link Addresses.
- STOPIO- stop Data I/O task on all or selected Link Addresses, and remove from Active Task List.
- XFRTBL- transfer selected LM system tables to/from LMG.
- STATUS- request modification of SM status words, such as a status indication previously set by the LM.

DATA TRANSFER COMMANDS

- INIT - ~~interrupt~~ interrupt the LM, indicating the initiation of a data transfer.
- REL - interrupt the LM, releasing the I/O buffer, and indicating completion of the data transfer.

TABLE 4
SM STATUS AREA

The task responsible for setting each status below is given in brackets (all go through the SM Handler):

LM Function Status

- Status returned from last command, indicating success/failure and type of failure [LM Command Processor]

Data Transfer Status

- Acknowledgement of available buffer, which buffer available, buffer requested, which buffer taken, word subcount of message [LMG, Data I/O]

Status Alert

- Indicator to LMG of any revised status [any task setting status]

LA Status (for each LA)

- Operational Status: Up (fully functional)/ Not up (some problems)/down (non-functional) [Error Analysis]
- Asynchronous service request [Data I/O]

LM Hardware Status

- Operational Status: Up (fully functional) /Not up (some problems)/down (non-functional) [LM Diagnostic]
- Status returned from self-test of LM [LM Diagnostic]
- Status returned from test of ICA [LM Diagnostic]

NP Hardware Status

- Status of NP Interface Controller [NP Diagnostic]
- Status of NP's [NP Diagnostic]

LM Hardware Status, or NP Hardware Status. Both the LM Function Status and the Status Alert are written only by the LM and generate vectored interrupts to the LMG. The Data Transfer Status, however, does not generate any interrupts and may be modified by the LMG; its special nature is further described in Section 3.1.2.

The 128 words (256 bytes) of shared memory are addressed by the LMG using an 8-bit address. However, the 128 words seen by the LMG actually map to more than 128 words in the LM. In particular, the Data Transfer Status word and the 32-word Subsystem I/O Buffer are repeated 16 times in the LM, once for each Link Address (LA). An LA is assigned by an LM to each of its subsystems, and is used by the LMG when referring to a subsystem. Section 3.1.3 describes how the LMG specifies the LA during data transfer operations, thereby selecting the status word or I/O buffer being addressed. In addition, there are actually two Data Transfer command words in each LM, one for commands from the LMG's Manager Processor (MGP) and one for commands from the LMG's Link Control Unit (LCU), and each with its own interrupt vector.

Figure 11 illustrates how the 8-bit SM address combines with the LA to form the true memory address in the LM.

3.1.2 Data Transfer Operations

Figure 12 illustrates very generally how the SM is used during a data transfer. During LM initialization, each subsystem on an LM is given a Link Address (LA) which is used by the LM to refer to that subsystem and its associated tasks. When the LM goes through its con-

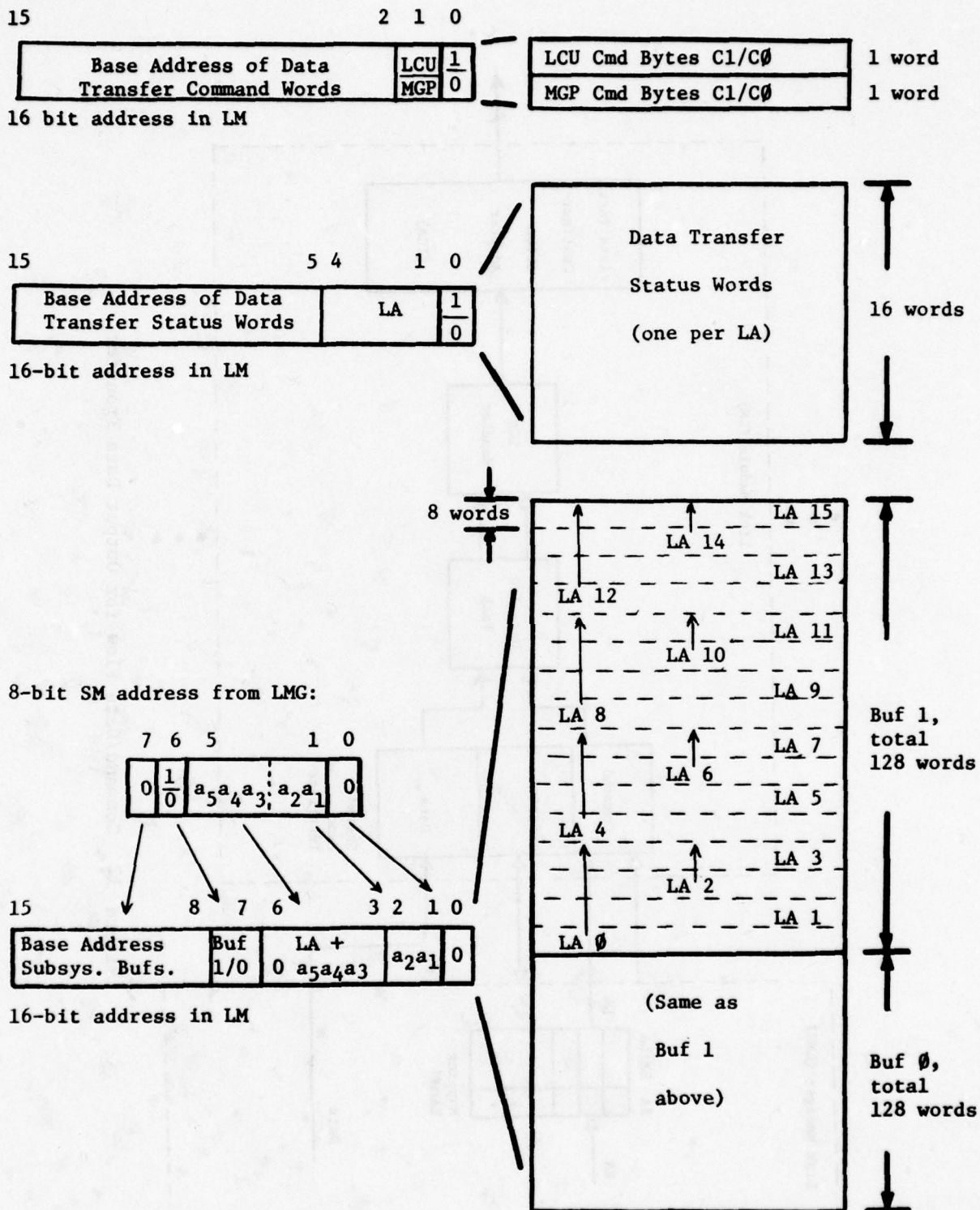


Figure 11 Mapping of Addresses in the LM

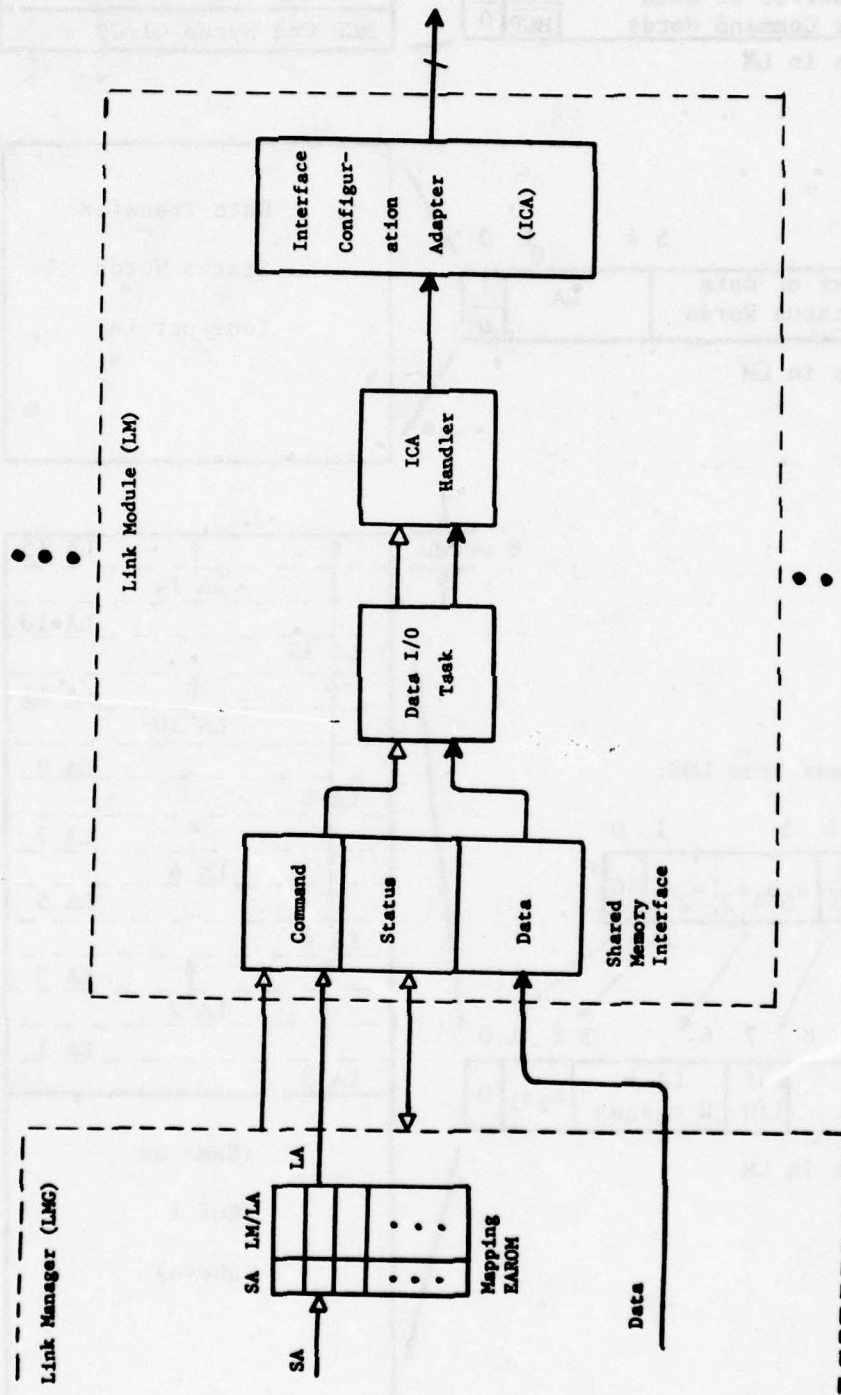


Figure 12 Command/Data Flow for Output Data Transfer

figuration process, as directed by a CONFIG command from the LMG, it assigns LA's to all subsystems and reports that LA back to the LMG. The LA within the Link Module, along with the Link Module number, together provide a unique LM/LA pair for each subsystem in the RLU. The LMG places the LM/LA address for each subsystem into an EAROM in order according to the subaddress (SA) which DAIS uses to address the subsystem. This entire initialization and configuration process is further described in Section 3.2.2 below.

When the RLU receives a write command, for example, on the 1553B bus, the SA is used to index down the EAROM to find the LM/LA of the destination subsystem. The LMG places the LA into the Data Transfer Command word in the shared memory of the correct LM. If the SM's Data Transfer Status indicates "ready", the output data is transferred to the SM Subsystem I/O Buffer. When all data has been written, the LMG writes "done" to the Data Transfer Command word, interrupting the LM. Section 3.1.3 below describes the handshaking protocol between LMG and LM in detail.

The LM then initiates the Data I/O task, along with the Data Conversion routines for that LA, to process the data and pass it to the ICA via the ICA Handler. Since the ICA has been previously configured properly for the destination subsystem, the data transfer should be successfully completed. If any problems are encountered by the Data I/O task such that the eventual data transfer is not successful, the LM so indicates in the Link Address Status area of the SM.

In the case of asynchronous messages from a serial input channel,

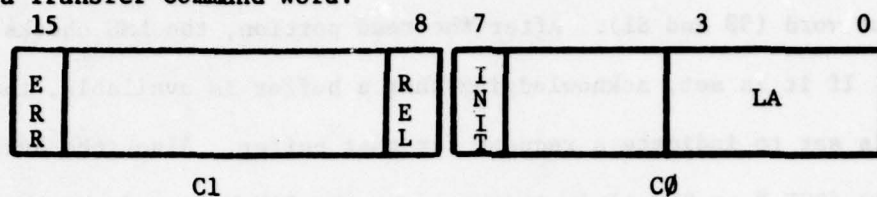
the LM must ask the LMG to make the asynchronous service request for the requesting LA. The Data I/O task first responds to the subsystem and places the data into the SM buffer. Then a bit is set in the LA Status word indicating a service request, and an interrupt to the LMG is generated through the Status Alert. The LMG makes the request to DAIS as described in Section 6. When the transmit command is eventually received from DAIS, the actual message transfer is handled no differently from other types of messages. The data buffer is held and the subsystem is locked out from further messages until a STATUS command is received from the LMG indicating successful receipt of the message by DAIS. The Data I/O task then uses the SM Handler and the ICA Handler to release the buffer, reset the bit in the LA Status, and lift the lock out to the subsystem.

3.1.3 Data Transfer Handshake

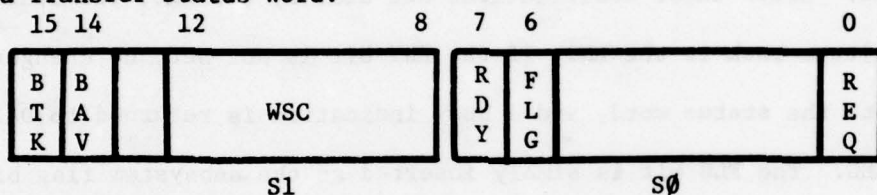
Figure 13 illustrates the specific command and status words involved in the data transfer handshake in shared memory between the LM's and the LMG. The handshake protocol and its implementation described here attempt to minimize the special hardware required at each LM's shared memory, putting a greater load on the LMP and software. When a prototype is actually built, software, hardware, speed, and other performance tradeoffs may result in a somewhat different protocol and implementation.

When the RLU receives a transmit or receive command on the 1553B bus, the SA is used to index down the EAROM to find the LM/LA of the source or destination subsystem. The LMG first writes the 4-bit LA

Data Transfer Command Word:



Data Transfer Status Word:



Data Transfer Handshake:

1. LMG sends C0 to LM, generating an interrupt

LA = 4-bit Link Address
 INIT = 1 to indicate initiation of message transfer

2. LMG performs read-modify-write on S1, S0
 FLG = 1 if subsystem is flagged as down
 RDY = 1 if a buffer is available
 BAV indicates which buffer (Buf 0/Buf 1)
 REQ = 1 to take the buffer
 BTK indicates which buffer is taken
 WSC = word subcount, no. of words being transferred

3. LMG transfers data

4. LMG sends C1 to LM, generating an interrupt

REL = 1 to release buffer and interrupt LM
 ERR = 1 if an error was encountered

Figure 13 Data Transfer Handshake Protocol

into the LM's data transfer command byte zero (C0) with the INIT bit set, causing an interrupt and notifying the LM of the initiation of a data transfer. Next the LMG does a read-modify-write operation on the entire status word (S0 and S1). After the read portion, the LMG checks the RDY bit. If it is set, acknowledging that a buffer is available, the REQ bit is set to indicate a request for that buffer. Also, the available buffer (BUF 0 or BUF 1) is indicated by the BAV bit, and the LMG will set the BTK bit equal to BAV to record which buffer has been taken. Finally the WSC field is set equal to the number of words to be transferred. After these modifications are done by the LMG, the status word is written back to the LM. If the RDY bit is not set, no changes are made to the status word, and a busy indication is returned to DAIS by the LMG. The FLG bit is simply inserted as the subsystem flag bit in the 1553B status word.

The data transfer now takes place, with the LMG addressing either buffer 0 or buffer 1, according to which was available and taken. After the transfers to all LM's and LA's participating in that message are complete, the LMG will write command word one (C1) to the LM's. Doing so will automatically cause a completion interrupt to the LMP.

The ERR bit, if set, indicates that, although the buffers are released, the correct data has not yet been transferred. The LMP will then clear only the REQ bit for each LA participating in the exchange. If the ERR bit is clear, the LMP will also clear the RDY bit for each such LA, and will proceed to set up the buffers and status bytes (S0 and S1) for the next transfer, either immediately or as they become

available. An exception is asynchronous subsystems. The buffers for them are held until a STATUS command is received from the LMG to release them and clear the RDY bit.

The data transfer status word was accessed by the LMG in a read-modify-write operation so that the LM cannot access the same word between separate read and write operations. The LM must be locked out during this period. Likewise, special hardware will be build into the SM so that the LMP can hold off LMG accesses to SM when the LMP wishes to assess the status word. To the LMG this action will appear simply as a slow response from the SM, and not as a busy subsystem. The exact combination of software and hardware interlocks to allow proper access by both sides to the status word must be carefully worked out during design of the prototype.

3.2 LINK MODULE INTERNAL ARCHITECTURE

3.2.1 Organization

Operations within the LM are directed by a real-time executive which has as its primary functions task scheduling, task completion analysis, and interrupt processing. Most other system functions are handled by individual system tasks. Table 5 gives details on all system tasks and programs, and Table 6 describes the system tables constructed by and referenced by these tasks and the executive. Appendix C provides more detail on all facets of the system software.

Two of the tasks/programs in Table 5 are special in that they are not native to the LM, but are either extracted from the subsystem nameplates, or are downloaded from the LMG using the PRGMLD command

TABLE 5
SYSTEM TASKS

<u>NAME</u>	<u>FUNCTIONS</u>	<u>INVOKED BY</u>
NP Initialization	<ul style="list-style-type: none"> - initialize all NP's - construct NP Topology Table (NPT) 	<ul style="list-style-type: none"> - NPINIT command
NP Diagnostic	<ul style="list-style-type: none"> - perform diagnostics on SIC and/or selected NP's - report hardware status of NIC and NP's 	<ul style="list-style-type: none"> - NPDIA command - Error Analysis Task
LM Diagnostic	<ul style="list-style-type: none"> - perform diagnostics on LM processor, memory, and/or ICA - report hardware status of LM and ICA 	<ul style="list-style-type: none"> - LMDIAG command - Error Analysis Task
Configuration	<ul style="list-style-type: none"> - configure ICA for each subsystem - construct Subsystem Configuration Table (SCT) 	<ul style="list-style-type: none"> - CONFIG command
Program Load	<ul style="list-style-type: none"> - load Data Conversion programs and Subsystem Diagnostic programs from NP or LMG - construct Subsystem Program Directory (SPD) 	<ul style="list-style-type: none"> - PRGMLOAD command
LM Command Processor	<ul style="list-style-type: none"> - analyze function commands to LM from LMG and invoke corresponding tasks - report command completion status to LMG 	<ul style="list-style-type: none"> - receipt of a command from LMG
Data I/O	<ul style="list-style-type: none"> - transfer data between subsystem & LMG - call Data Conversion program to process the data 	<ul style="list-style-type: none"> - WRITE command (output) - LM timer or external interrupt (input)
Data Conversion	<ul style="list-style-type: none"> - convert and check subsystem data - report status of data to Data I/O task 	<ul style="list-style-type: none"> - Data I/O task
Subsystem Diagnostic	<ul style="list-style-type: none"> - perform diagnostics on subsystem - analyze errors returned to Data I/O task by Data Conversion Program - write error log to NP - report Link Address operational status 	<ul style="list-style-type: none"> - SSDIAG command - Error Analysis Task

TABLE 5
SYSTEM TASKS (CONT'D.)

<u>NAME</u>	<u>FUNCTIONS</u>	<u>INVOKED BY</u>
Error Analysis	<ul style="list-style-type: none"> - analyze non-zero completion status from tasks - invoke appropriate diagnostic tasks - maintain an error log in EAROM 	<ul style="list-style-type: none"> - EXEC (upon task completion with non-zero status)
Table Transfer	<ul style="list-style-type: none"> - transfer system tables to/from LMG 	<ul style="list-style-type: none"> - XFRTBL command
ICA Handler	<ul style="list-style-type: none"> - transfer data to/from ICA channels - report ICA hardware status 	<ul style="list-style-type: none"> - Diagnostic task - Configuration task - Data I/O task - Subsystem Diagnostic task
SM Handler	<ul style="list-style-type: none"> - transfer data to/from Shared Memory 	<ul style="list-style-type: none"> - Data I/O task - Error Analysis task - LM Command Processor task
NP Handler	<ul style="list-style-type: none"> - transfer data to/from NP's - report status returned from NP 	<ul style="list-style-type: none"> - NP Initialization task - NP Diagnostic task - Program Load task - Subsystem Diagnostic task - Configuration task

TABLE 6
SYSTEM TABLES

<u>NAME</u>	<u>CONTENTS</u>	<u>AUTHOR TASK</u>	<u>READER TASKS</u>
NP Topology Table (NPT)	By Level: NP address, NP configuration EID configuration	NP Initialization	Through NP Handler: Configuration, NP Diagnostic
Subsystem Configuration Table (SCT)	By Link Address: ICA configuration and its source (NP or LMG), NP address, EID	Configuration	Through ICA Handler: Data I/O, Subsystem Diagnostic, LM Diagnostic Through NP Handler: Subsystem Diagnostic, NP Diagnostic, Program Load
Subsystem Program Directory (SPD)	By Link Address: pointer to data con- version program, its source (NP or LMG); pointer to Subsystem Diagnostic program, its source (NP or LMG)	Program Load	EXEC
System Task Directory (STD)	By Name: pointers to all system tasks	-	EXEC
Active Task List (ATL)	Linked list of all active tasks, point- ers to them, task states	EXEC	EXEC

and Program Load task. These two are the Data Conversion Program and the Subsystem Diagnostic Task, and are unique to each subsystem. The Data Conversion Program is called by the Data I/O Task for each LA to perform the data conversion and data validity checks for that LA. The Subsystem Diagnostic Task is invoked by the Error Analysis Task or by an SSDIAG command from the LMG, to exercise a subsystem and check for proper functioning.

Since these two programs are written unique to each subsystem, most likely by the subsystem manufacturer, they are written in a standard high level interpretive language. The executive, therefore, incorporates an interpreter which executes these programs when they are active. Section 7 provides more information on the nature of these programs.

3.2.2 Initialization and Configuration

Although the system tasks may be invoked at any time, most of the tasks are invoked during the standard LM initialization sequence. The sequence is thoroughly outlined in Figure 14 . In general, the LM obtains from each NP the information necessary to configure the subsystems. The subsystem's EID, extracted from the NP and also previously known to DAIS, is the common identifying tag allowing the LMG to make the correspondence between the DAIS-assigned subaddress (SA) and the LM-assigned Link Address (LA). This correspondence is stored in the LMG's mapping EAROM (see Figure 12) so that data transfers to/ from the subsystems are correctly routed.

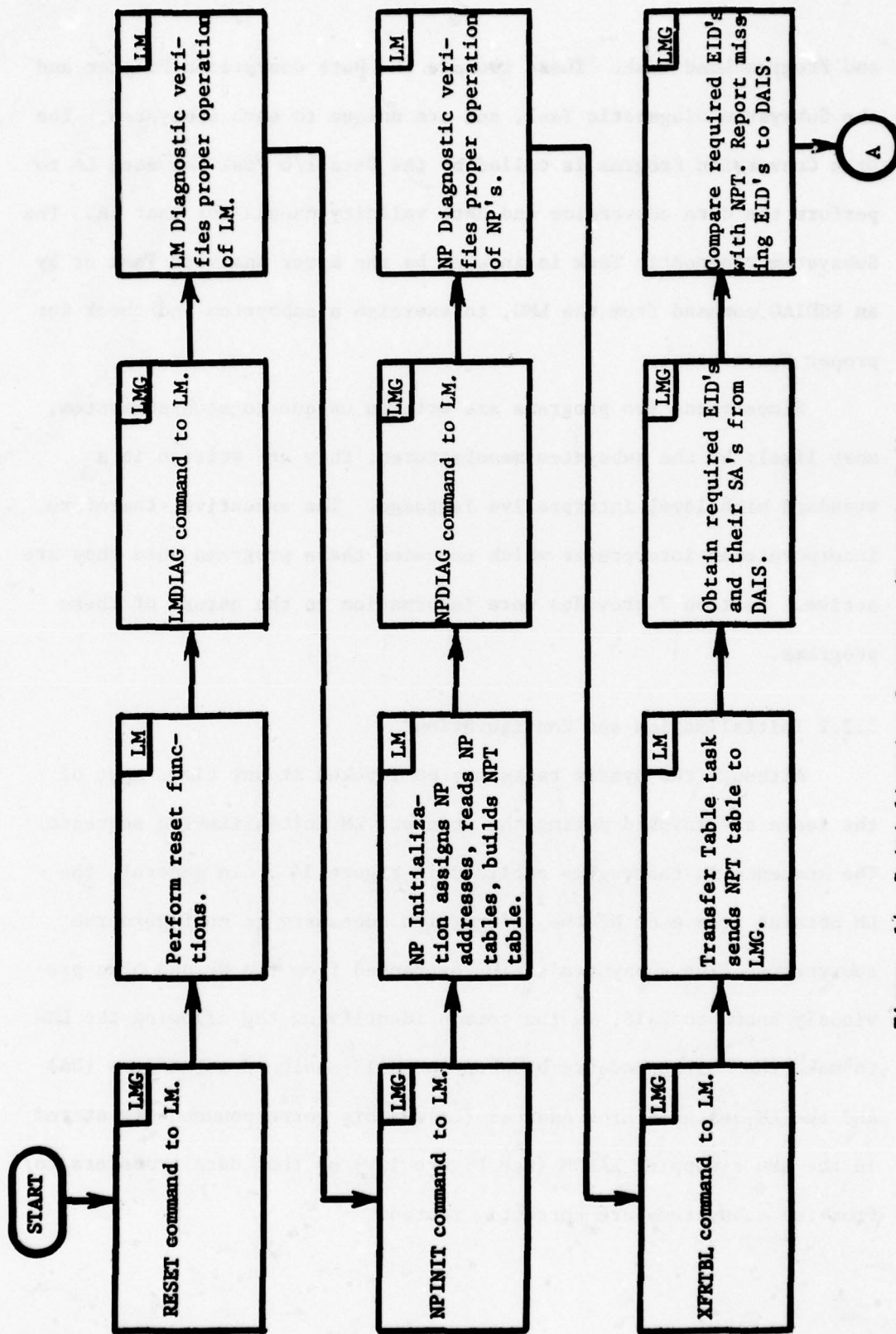


Figure 14 Initialization Sequence (Part 1 of 2)

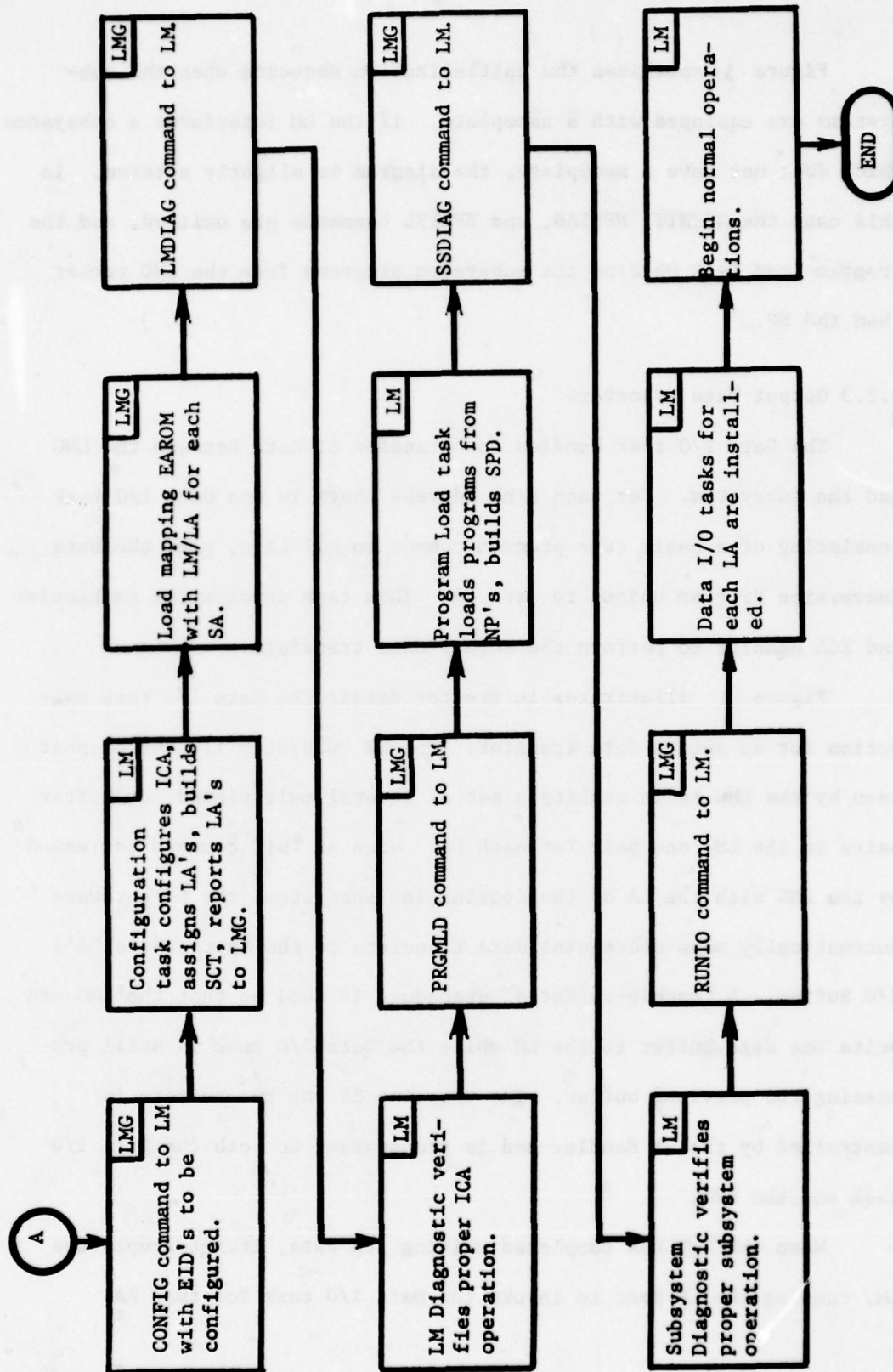


Figure 14 Initialization Sequence (Part 2 of 2)

Figure 14 outlines the initialization sequence when the subsystems are equipped with a nameplate. If the LM interfaces a subsystem which does not have a nameplate, the diagram is slightly altered. In this case the NPINIT, NPDIAG, and XFRTBL commands are omitted, and the Program Load Task Obtains the subsystem programs from the LMG rather than the NP.

3.2.3 Output Data Transfers

The Data I/O task handles the transfer of data between the LMG and the subsystem. For each Link Address there is one Data I/O task consisting of a basic core program common to all LA's, plus the Data Conversion Program unique to each LA. This task invokes the SM Handler and ICA Handler to perform the actual data transfers.

Figure 15 illustrates in greater detail the Data I/O task execution for an output data transfer. The SM Subsystem I/O Buffer pair seen by the LMG is in reality a set of several multiplexed out buffer pairs in the LM, one pair for each LA. When an INIT command is issued by the LMG with the LA of the destination subsystem, the SM hardware automatically maps subsequent data transfers to the appropriate LA's I/O Buffer. A "double-buffered" procedure is used so that the LMG can write one data buffer to the LM while the Data I/O task is still processing the previous buffer. The toggling of the two buffers is controlled by the SM Handler and is transparent to both the Data I/O task and the LMG.

When the LMG has completed writing its data, it interrupts the LM, causing the LM Exec to invoke the Data I/O task for that LA.

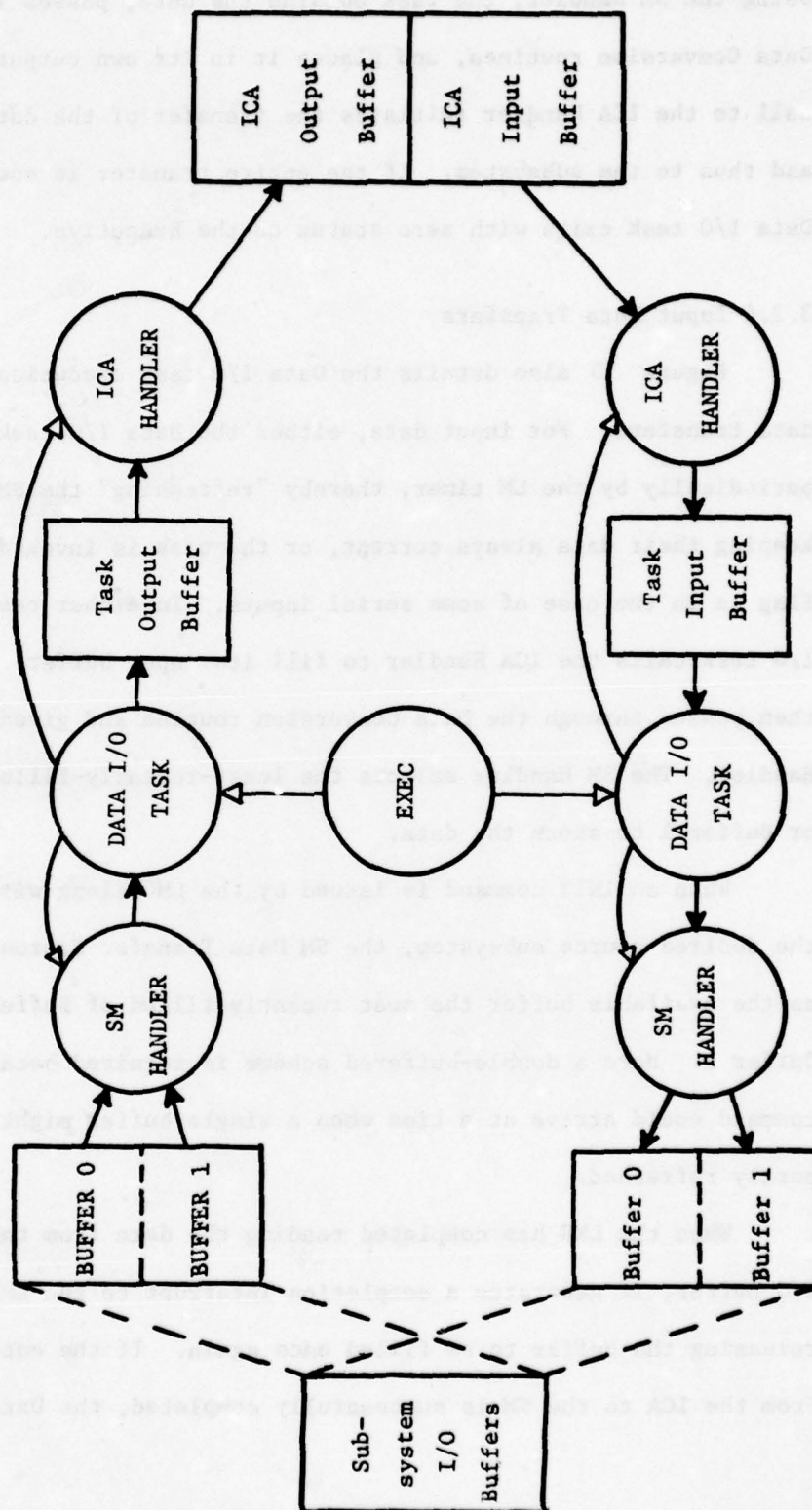


Figure 15 Data I/O Task Execution

Using the SM Handler, the task obtains the data, passes it through its Data Conversion routines, and places it in its own output buffer. A call to the ICA Handler initiates the transfer of the data to the ICA, and thus to the subsystem. If the entire transfer is successful, the Data I/O task exits with zero status to the Executive.

3.2.4 Input Data Transfers

Figure 15 also details the Data I/O task execution for input data transfers. For input data, either the Data I/O task is invoked periodically by the LM timer, thereby "refreshing" the SM buffers and keeping their data always current, or the task is invoked by an external flag as in the case of some serial inputs. In either case, the Data I/O task calls the ICA Handler to fill its input buffer. The data is then passed through the Data Conversion routine and given to the SM Handler. The SM Handler selects the least-recently-filled of Buffer 0 or Buffer 1 to store the data.

When an INIT command is issued by the LMG along with the LA of the desired source subsystem, the SM Data Transfer Status word indicates as the available buffer the most recently filled of Buffer 0 and Buffer 1. Here a double-buffered scheme is required because an INIT command could arrive at a time when a single buffer might be only partly refreshed.

When the LMG has completed reading the data from the Subsystem I/O Buffer, it generates a completion interrupt to the LM, thereby releasing the buffer to be filled once again. If the entire transfer from the ICA to the SM is successfully completed, the Data I/O task

exits with zero status to the Executive.

3.2.5 Error Processing

While the Data I/O task is active, errors can occur at a number of points. If the ICA Handler encounters an error which its own retries could not correct, it sets the appropriate LM Hardware Status in Shared Memory and reports the error to the Data I/O task which invoked it. Similarly, if the data validity checks in the Data Conversion program discover an error, the error is reported to the Data I/O task. Any such error conditions result in a non-zero status returned to the Executive when the task exits. The status is sufficiently detailed to indicate what action should be taken.

When the Exec observes the non-zero completion status, it removes the Data I/O task from the Active Task List and activates the Error Analysis task. The procedure is illustrated in Figure 16 . As indicated by the completion status, the Error Analysis task invokes the NP Diagnostic, the LM Diagnostic, and/or the Subsystem Diagnostic for that LA. The results obtained from these diagnostic tasks are then analyzed to determine the future status of that LA. The LA Operational Status in SM could be revised to indicate that the LA is "up" (fully functional), "not up" (functional, but without 100% reliability), or "down" (no longer functional). The LMG, alerted to the revised status, can rerun the diagnostics if it wishes, or take whatever additional error processing steps the situation warrants.

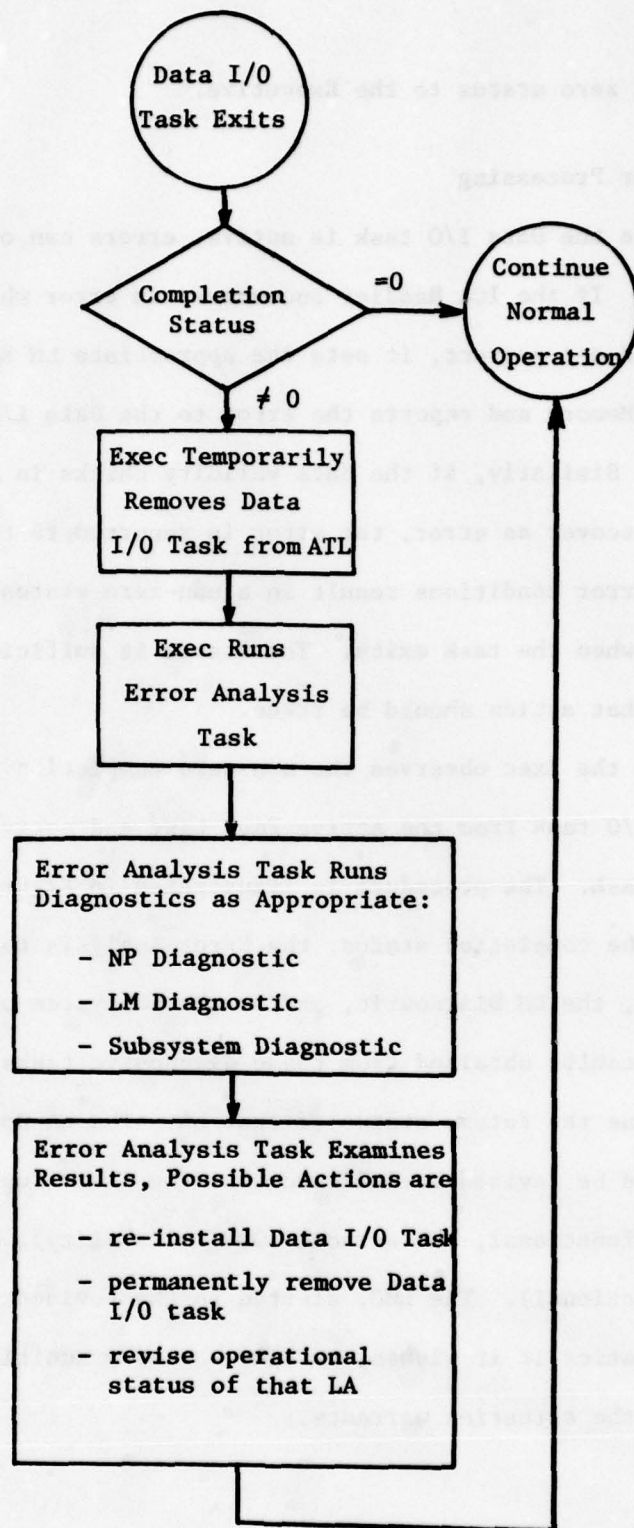


Figure 16 LM Error Processing

3.3 LINK MODULE TESTING

The entire LM is designed so as to be easy to test for proper operation. The LM Diagnostic is capable of performing comprehensive tests of the processor instructions, memory, and I/O interfaces, including the ICA. In addition, the NP Diagnostic can fully test the NIC. Each of these tasks can be initiated by external command from the LMG, or some other suitable testing computer.

In addition, the shared memory interface to the LM is a simple way of accessing all commands, status, and data. For testing purposes, the LMG can be replaced by an operator panel capable of accessing the shared memory. Thus the LM Function Commands can be entered manually to exercise the LM. Status can be observed by the operator, and data can be manually transferred to and from the SM Data Buffers.

SECTION IV

INTERFACE CONFIGURATION ADAPTER

The interface Configuration Adapter (ICA) is a link between the internal bus of the Link Module (LM) and the Subsystem Data Channel (SDC). As such, its principal function is conversion of information between the digital form on the internal bus and the discrete, analog, and serial forms on the SDC. Configuration of interface hardware, and therefore of the types of signals on the SDC, is soft programmable via information provided to the ICA through the internal bus. The ICA has the following general features:

- It supports the simultaneous operation of four interface groups.
- Each group comprises four channels which may be configured to perform together any of the I/O functions listed in Table 4-1.
- The ICA interface is compatible with both signal oriented and function oriented subsystem interfaces.
- Interface groups may be operated together to support a signal type interface with 4, 8, 12, or 16 channels.
- Control of the ICA groups is done by setting configuration parameters and issuing a control command.
- The high and low levels of binary output signals are programmable.
- The threshold level for discrimination between the levels of binary input signals is programmable.
- Testability is an inherent feature of the ICA design.
- Each output signal is monitored continuously.
- Each output can be wrapped around to test the corresponding input channel.
- The ICA can be used as the subsystem interface for link modules or it can be incorporated into interface modules of the current type.

TABLE 7
ICA INTERFACE TO SUBSYSTEMS

SIGNAL INTERFACE	NUMBER OF CHANNELS PER ICA GROUP
<u>Discrete Input:</u>	
Differential	4
Single-ended	4
Switch closure	4
Momentary Open/Ground	4
<u>Analog Input:</u>	
Differential DC	4
Single-ended DC	4
Differential AC	4
Single-ended AC	4
Synchro	3
<u>Discrete Output:</u>	
Differential	4
Single-ended	4
Switch closure	0
<u>Analog Output:</u>	
Differential DC	4
Single-ended DC	4
Differential AC	4
Single-ended AC	4
Synchro	
<u>Serial:</u>	
Serial Input	1
Serial Output	1

4.1 ARCHITECTURE

Architecturally, as shown in Figure 17 , the ICA has four major blocks: the interface buffers, the signal interface, data conversion circuits, and control logic. Its operation is described in terms of the data flow between the interface buffer and the signal interface (Section 4.2) and by its control logic and internal timing signals (Section 4.3).

Except for LMP interrupts initiated by the ICA, all data, control, and configuration information passing between the LMP and ICA is routed through the shared interface buffer, which appears as 128 16-bit words in the memory address space of the LMP. This buffer is divided into five functional areas, in which are stored, respectively:

- configuration parameters,
- operational control commands,
- operational status information,
- input data and status information,
- output data and status information.

The following signals, which should be thought of as passing through the LMP/ICA interface, are required for operation of the ICA:

- master clock,
- memory interlace clock,
- interface buffer address,
- interface read/write,
- AC synchronization clock and phase,
- LMP interrupt request.

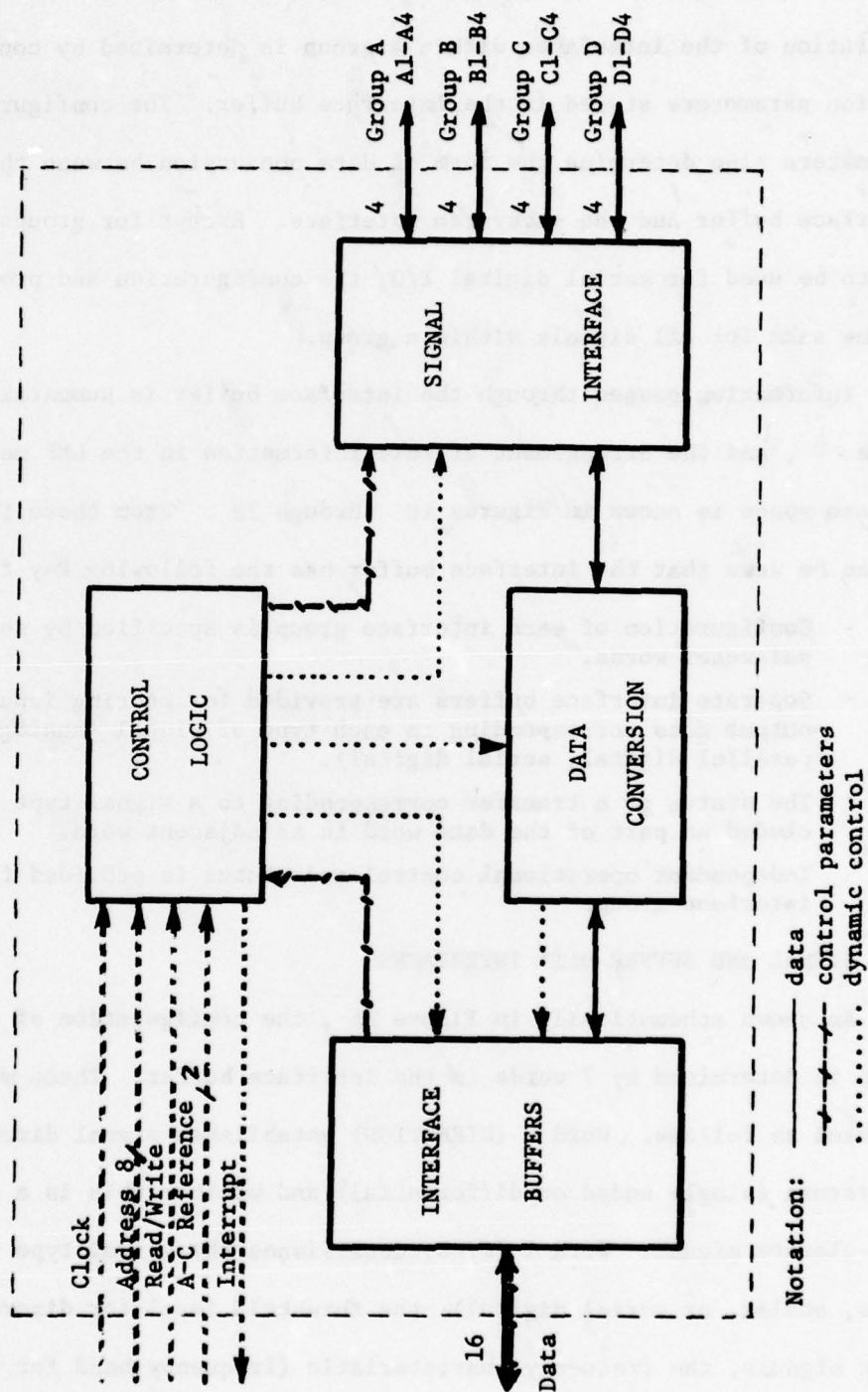


Figure 17 Internal Organization of the Interface Configuration Adaptor (ICA)

On the other side of the ICA, the signal interface provides 16 channels organized as four groups each with four channels. The configuration of the interfaces within a group is determined by configuration parameters stored in the interface buffer. The configuration parameters also determine the form of data conversion between the interface buffer and the subsystem interface. Except for groups which are to be used for serial digital I/O, the configuration and processing is the same for all signals within a group.

Information passed through the interface buffer is summarized in Table 8 , and the arrangement of this information in the LMP memory address space is shown in Figures 18 through 20 . From these figures it can be seen that the interface buffer has the following key features:

- Configuration of each interface group is specified by seven parameter words.
- Separate interface buffers are provided for storing input and output data corresponding to each type of signal (analog, parallel digital, serial digital).
- The status of a transfer corresponding to a signal type is included as part of the data word in an adjacent word.
- Independent operational control and status is provided for each interface group.

4.2 SIGNAL AND BUFFER DATA INTERFACES

As shown schematically in Figure 21 , the configuration of each group is determined by 7 words in the interface buffer. These words are used as follows. Word 0 (DIRECTION) establishes signal direction and return (single ended or differential) and whether this is a contact-closure signal. Word 1 (TYPE) establishes the signal type (discrete, analog, or serial digital), the threshold level for discrete input signals, the frequency characteristic (frequency band for the filters, and logical processing of momentary signals). Word 2 (LEVEL)

TABLE 8
ICA INTERFACE TO THE LINK MODULE

CONFIGURATION PARAMETERS (8 words/group)

- Direction (in/out), signal return (common, differential), contact closure
- Type (serial digital, parallel digital, analog) frequency (filter, momentary)
- Levels (2 words) (high level, low level, threshold)
- Scan (flag/refresh, AC/DC)
- Refresh rate
- Transmission/reception rate
- Word count

OPERATIONAL CONTROL (5 words)

- One word/group and one word for all groups
- Commands
 - Reset
 - Test
 - Test Halt
 - Run
 - Halt

OPERATIONAL STATUS (5 words)

- Group (mode of operation, subsystem error, ICA hardware: A/D, UART)
- ICA

INPUT DATA AND STATUS (30 words)

- Analog: one 16-bit word/channel (12 bit data, 4 bit status: overflow, S/H time out)
- Serial digital: two 16-bit word/channel (one data, one status: parity, framing, overrun, time out, buffer full)
- Parallel digital: one 16-bit word/group (4 bit data, 4 bit status), two 16-bit word/all groups (16 bit data, 16 bit status) (momentary closure sets buffer full, clear w/read)

OUTPUT DATA AND STATUS (29 words)

- Analog: one 16-bit word/channel
- Serial digital: two 16-bit word/channel (one data, one status: parity, time out buffer empty)
- Parallel digital: one 16-bit word/group (4 bit data), one 16-bit word/all groups

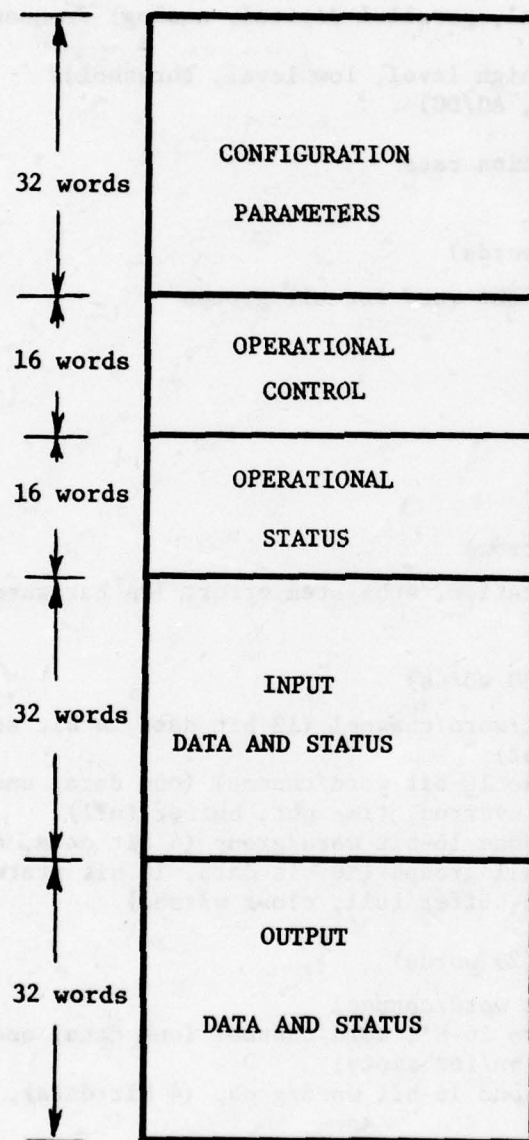


Figure 18 Address Allocation for ICA
Interface Buffers

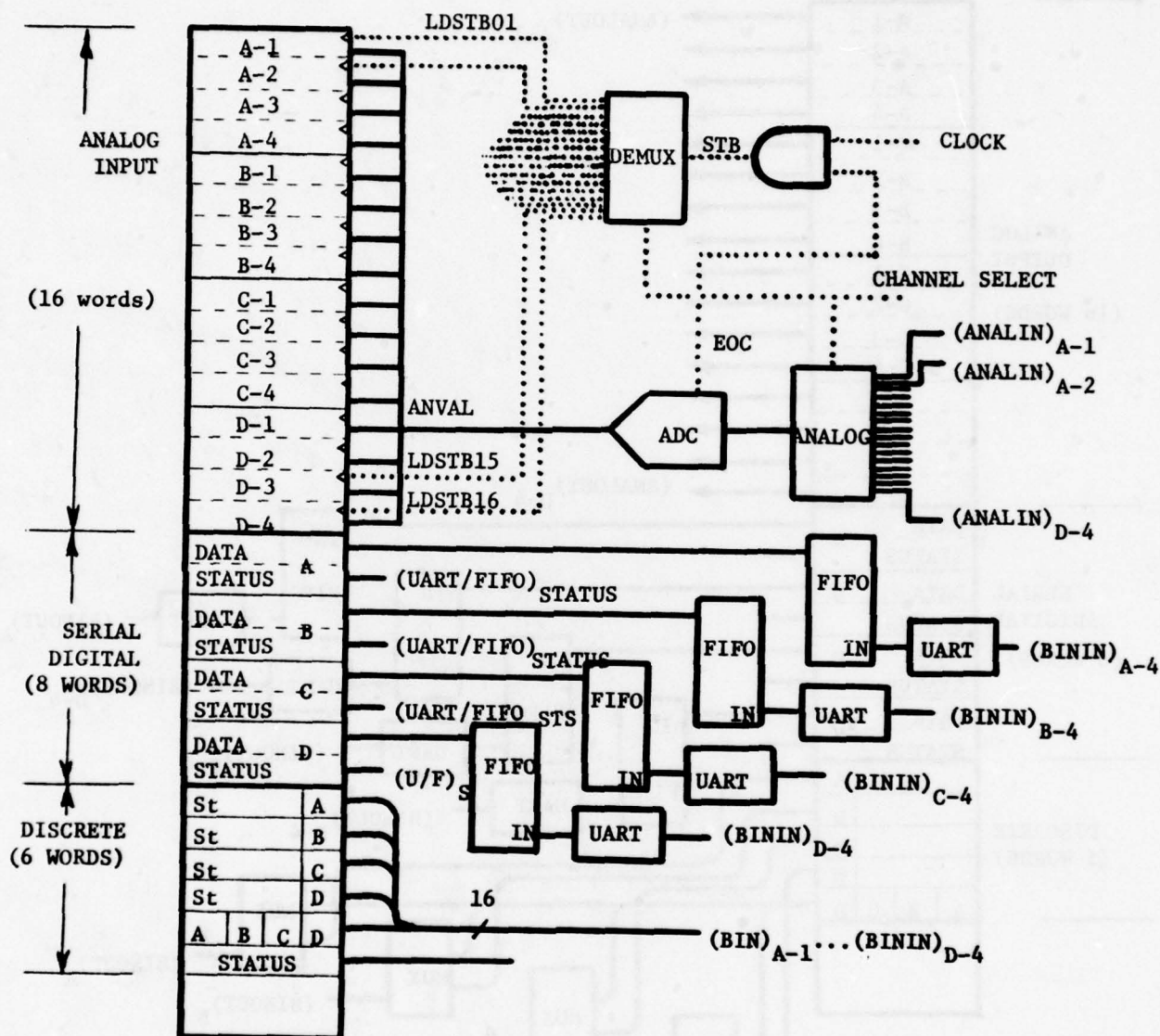


Figure 19 Organization of the Data Input (DATAIN)

Section of the Interface Buffer

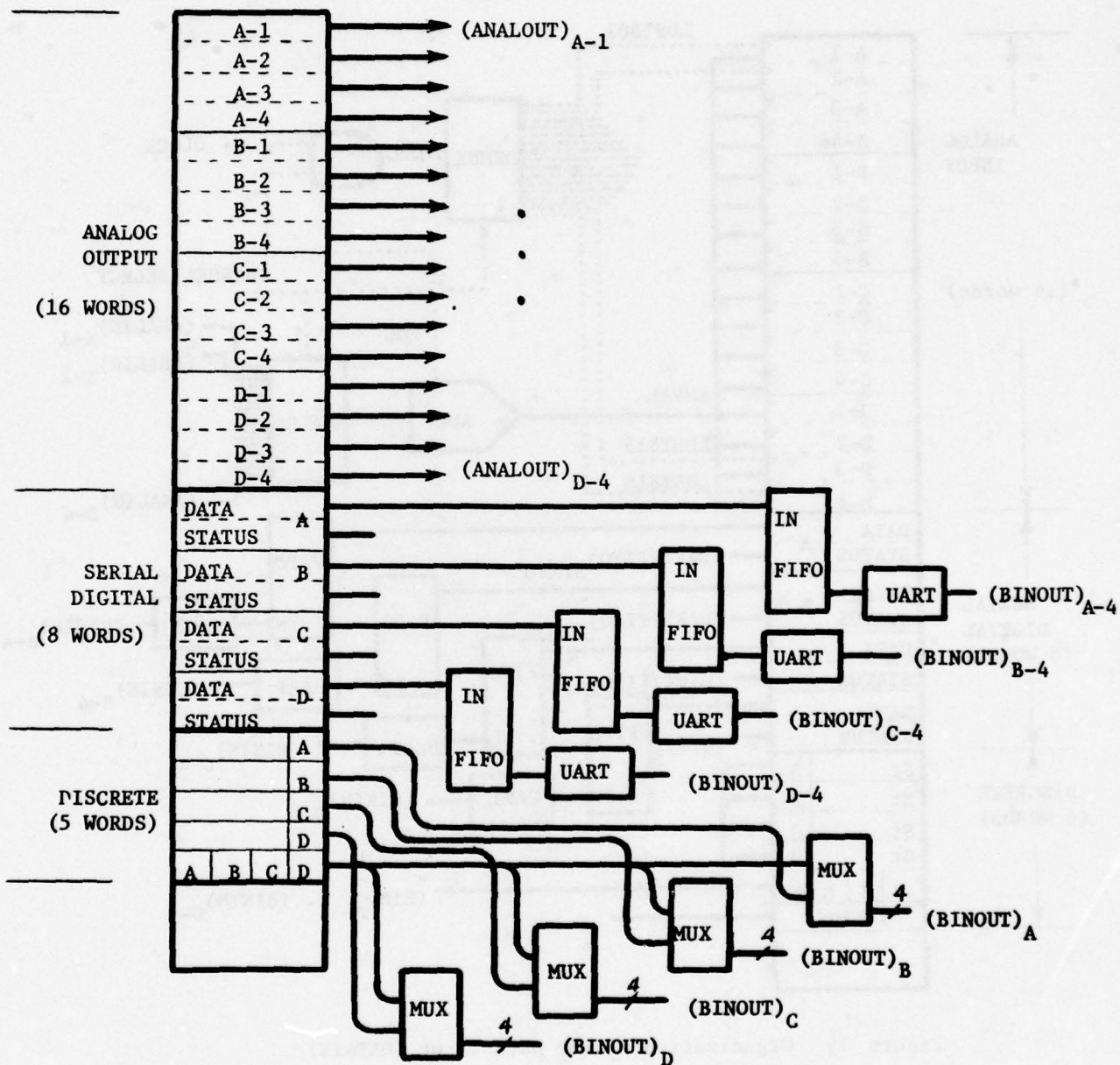


Figure 20 Organization of the Data Output (DATAOUT)
Section of the Interface Buffer

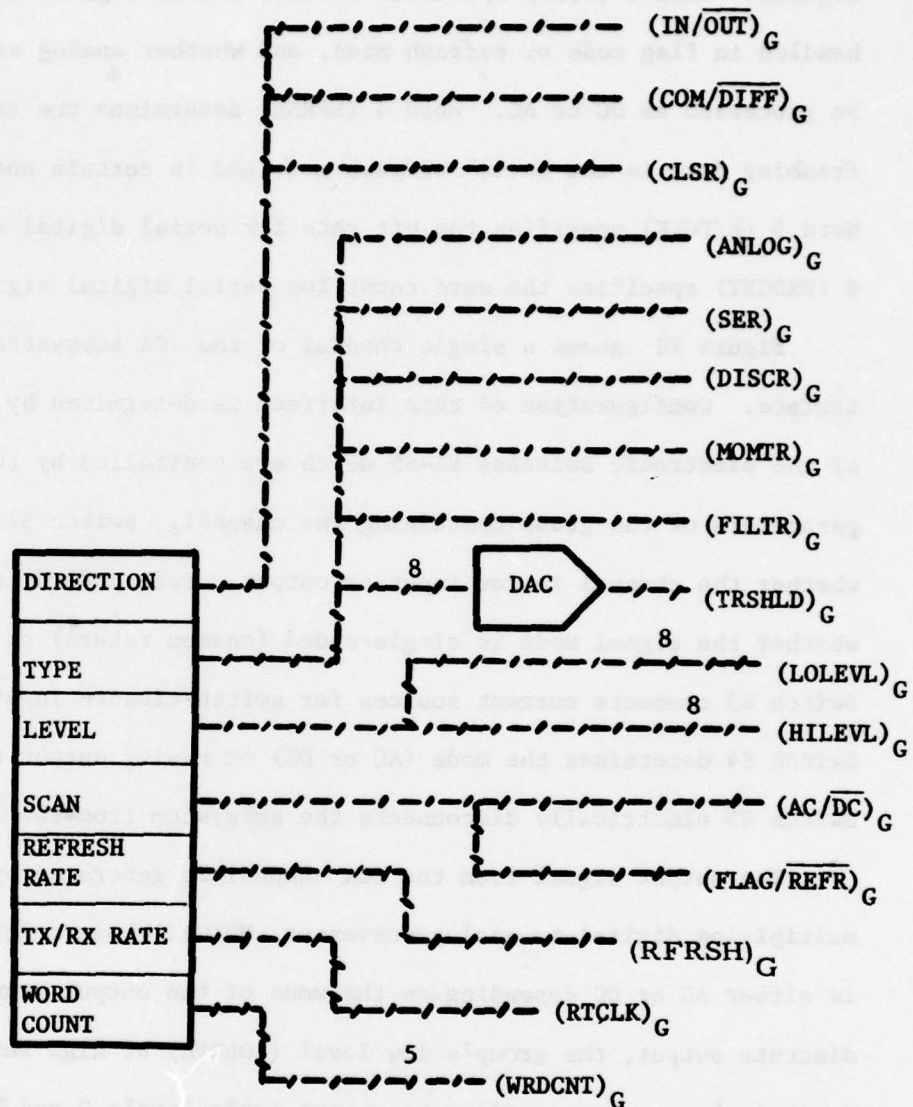


Figure 21 Organization of Parameter Section of the Interface Buffer

specifies the signal level or levels for generating discrete output signals. Word 3 (SCAN) specifies whether serial signals are to be handled in flag mode or refresh mode, and whether analog signals are to be processed as DC or AC. Word 4 (RFRSH) determines the rate of refreshing data in the serial refresh mode and in certain analog modes. Word 5 (R/TCLK) specifies the bit rate for serial digital signals. Word 6 (WRDCNT) specifies the word count for serial digital signals.

Figure 22 shows a single channel of the ICA subsystem signal interface. Configuration of this interface is determined by the setting of the electronic switches S1-S5 which are controlled by the configuration parameters of the group containing the channel. Switch S1 determines whether the channel is for input or output. Switch S2 determines whether the signal mode is single-ended (common return) or differential. Switch S3 connects current sources for switch-closure input signals. Switch S4 determines the mode (AC or DC) of analog output signals. Switch S5 electrically disconnects the subsystem from the ICA.

The output signal from the ICA channel is generated by the 12-bit multiplying digital-to-analog converter (MDAC), whose reference input is either AC or DC depending on the mode of the output signal. For discrete output, the group's low level (LOLEVL) or high level (HILEVL) digital value, corresponding to output logic levels 0 and 1, is gated through a digital multiplexer to the MDAC. Digital values for analog output (ANALOUT) are gated through a tri-state buffer to the MDAC.

Input signals are brought through an instrumentation amplifier and digitally controlled filter to a sample-and-hold circuit (S/H) for analog input, and to a comparator (COMP) for discrete input. Threshold for discrete input is established by the group threshold (TRSHLD) value.

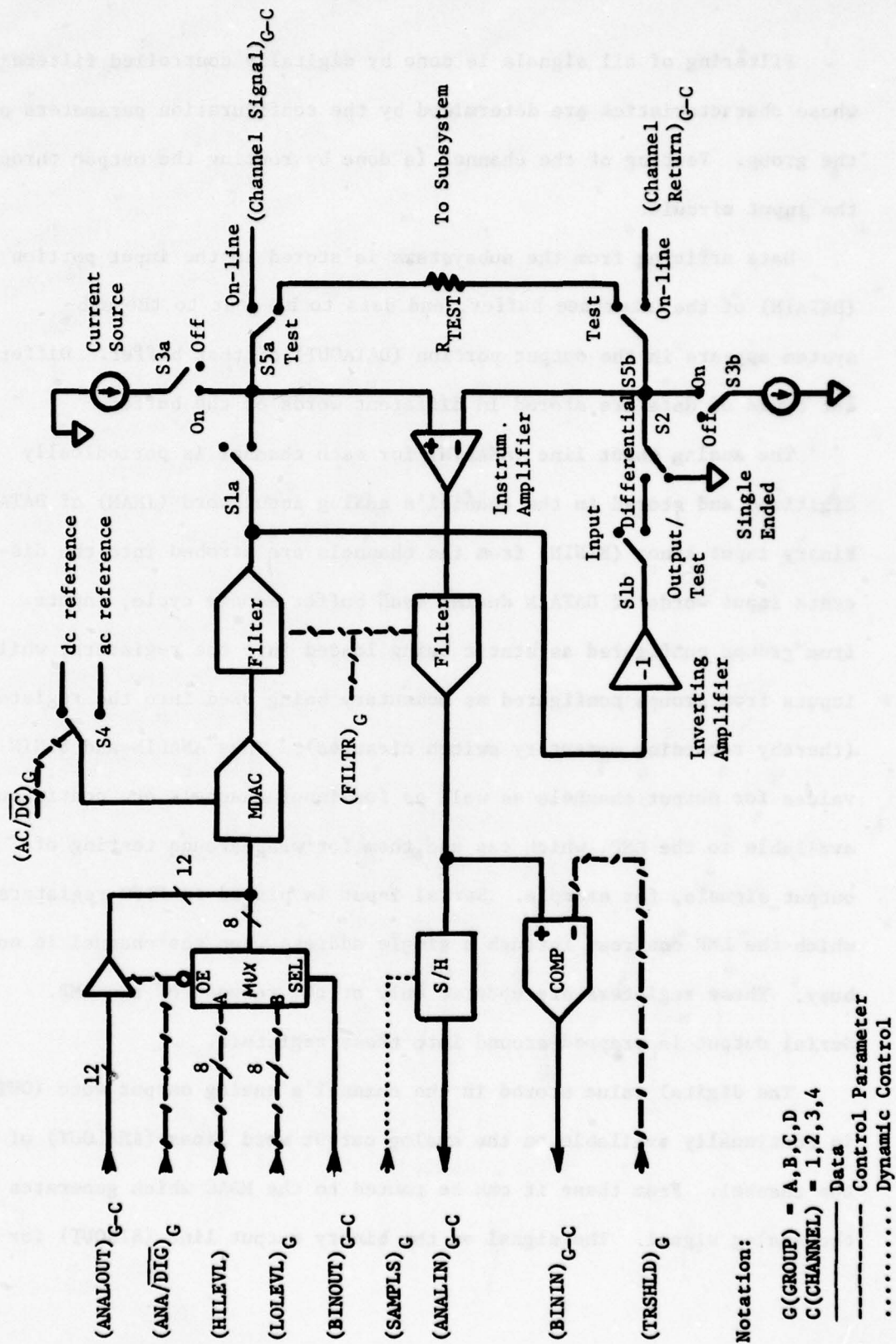


Figure 22 ICA Signal Interface

Filtering of all signals is done by digitally controlled filters whose characteristics are determined by the configuration parameters of the group. Testing of the channel is done by routing the output through the input circuit.

Data arriving from the subsystems is stored in the input portion (DATAIN) of the interface buffer, and data to be sent to the subsystem appears in the output portion (DATAOUT) of that buffer. Different types of data are stored in different words of the buffer.

The analog input line (ANALIN) for each channel is periodically digitized and stored in the channel's analog input word (INAN) of DATAIN. Binary input lines (BININ) from the channels are strobed into the discrete input words of DATAIN during each buffer memory cycle, inputs from groups configured as static being loaded into the registers, while inputs from groups configured as momentary being ORed into the registers (thereby recording momentary switch closures). Thus ANALIN and BININ values for output channels as well as for input channels are continuously available to the LMP, which can use them for wrap-around testing of output signals, for example. Serial input is placed in FIFO registers which the LMP can read through a single address when the channel is not busy. These registers are updated only at the request of the LMP. Serial output is wrapped-around into these registers.

The digital value stored in the channel's analog output word (OUTAN) is continually available on the analog output word lines (ANALOUT) of the channel. From these it can be routed to the MDAC which generates the analog signal. The signal on the binary output line (BINOUT) for

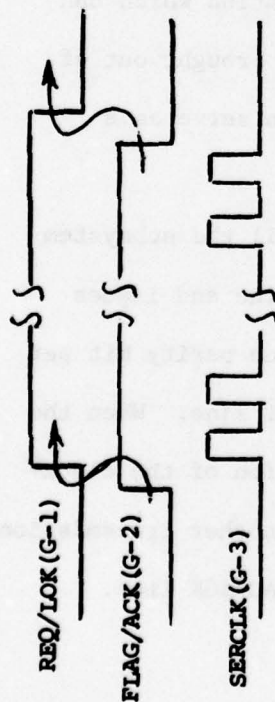
the channel carries the discrete output value (stored in OUTDIS) for the channel except when the channel is being used for serial output. In the latter case, it is generated by the serial control logic. Serial output data is stored in the channel's output FIFO register, which the LMP can load through a single address when the channel is not busy.

4.2.1 Serial Signals

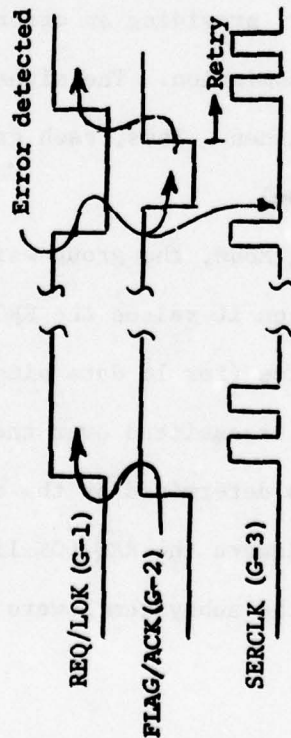
Figure 23 shows the timing for serial data transmission. The scheme shown differs from that of existing DAIS interface cards in that some of the signals have been combined to form a reduced set of signals: FLAG, ACKNOWLEDGE, and ERROR have been combined into FLAG/ACK, and REQUEST and LOCKOUT into REQ/LOK. Thus, serial channels require only four lines: FLAG/ACK, REQ/LOK, SERCLK (the clock line), and SERDAT (the data line). Output parity errors detected by the subsystem cause FLAG/ACK to drop immediately, providing an error indication which can be tested at the end of transmission. The signals are brought out of a group on the four signal lines. Thus, each group can serve as a serial input or output channel.

For input in the Flag Mode, the group waits until the subsystem raises the FLG/ACK line. Then it raises the REQ/ACK line and issues $(\text{word count}) \times 17$ clock pulses (for 16 data bits and one parity bit per word), and accepts the data transmitted over the SERDAT line. When the transmission is complete (as determined by the completion of the clock-pulse sequence), the group lowers the REQ/LOK line. Another transmission will not begin until after the subsystem lowers the FLAG/ACK line.

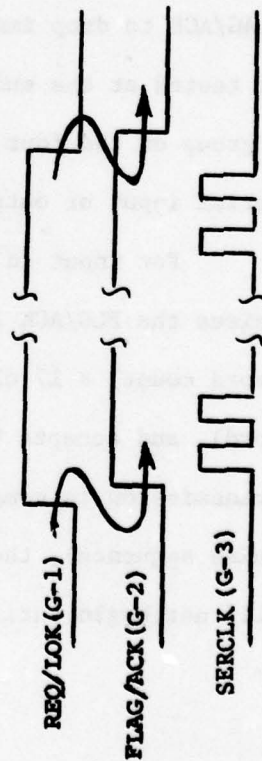
Interface Channel



(a) Input (Flag Mode w/no error detected)

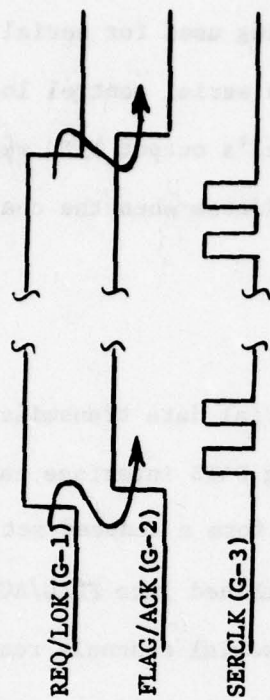


(b) Input (Flag Mode w/an error detected)

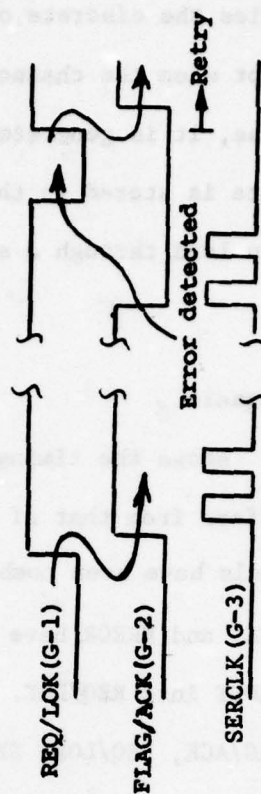


(c) Input (Refresh Mode)

Interface Channel



(d) Output (With No Error Detected)



(e) Output (With an Error Detected)

Figure 23 Handshake between LM and Subsystem for Serial Data Transmission

Input in the Refresh Mode is similar except that the transmission is initiated by the group's raising the REQ/LOK line. Clock pulses are not issued until after the subsystem acknowledges the request by raising the FLAG/ACK line. From this point on, operation is identical to that for Flag Mode.

Serial output operation is identical to serial input operation in the Refresh Mode, except that data is clocked out onto the SERDAT line by the group. Also, upon completion of data transmission, the group checks the state of FLAG/ACK before lowering the REQ/LOK line. If the state is low the subsystem detected a parity error in the transmission.

4.3 OPERATIONAL CONTROL AND TIMING

Operational states of an ICA group are shown in Figure 24 . The state of each group is determined by the configuration-parameters words and the operational control word (command) for the group. All groups are put into the UNASSIGNED state during power-on reset. A group is taken to the CONFIGURED state when a legitimate configuration is defined by the configuration-parameter words. The control word causes transitions between the CONFIGURED, TESTING, and RUNNING states. Input channels are electrically connected to the subsystems when CONFIGURED or RUNNING. Output channels are connected to the subsystems only in the RUNNING state. Thus, subsystems are unaffected by channels which are not in the RUNNING state.

Status information for each group appears in the status word for the group. Overall ICA status information, including summary information from the group status words appears in the ICA status word. The

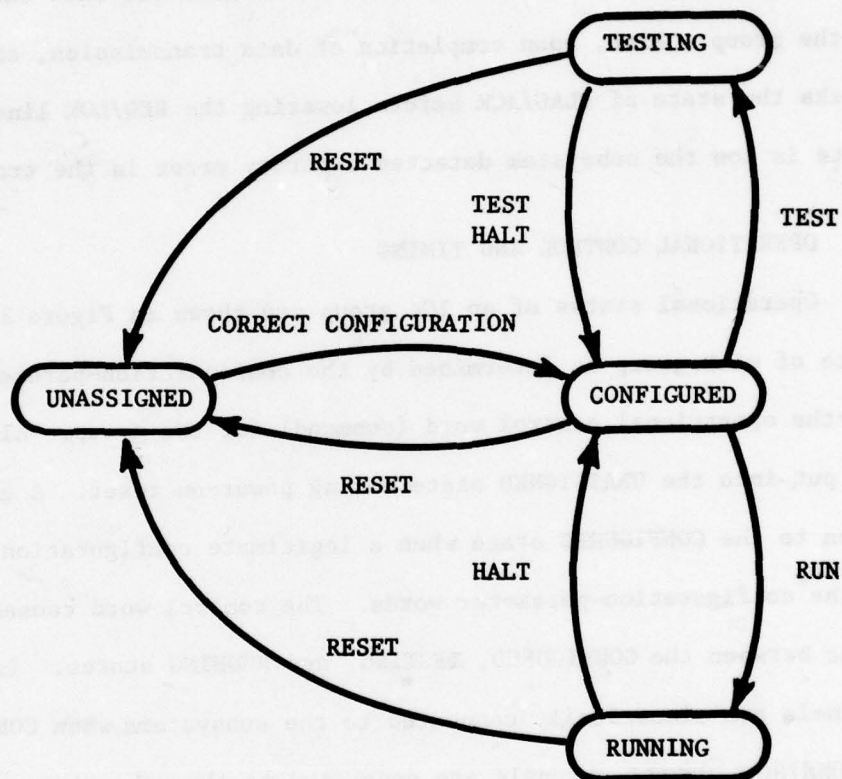


Figure 24 Operating States of Each ICA Group and Permissible Commands at Each State

LMP can examine this word to determine whether any of the groups need servicing.

Since the ICA operates in four states which are determined completely by the configuration and control words (and therefore not by any sequential operations), the basic control logic is quite simple, and the only timing problems relate to the operation of the subsystem interfaces themselves, and the interaction of these interfaces and the LMP interface.

The LMP interface is memory (some of which must be implemented with parallel-access static registers, of course) to which access by the ICA and the LMP is given in alternate hardware memory cycles. Thus, to each of the ICA and the LMP the interface appears to be memory which operates at half the actual memory speed of the hardware memory. In this section and subordinate sections only ICA memory accesses are under consideration so they will be timed to coincide with the ICA part of the memory cycle. The FIFO registers used for serial I/O are locked away from the LMP during the serial operations, so a group operating as a serial channel can access its FIFO registers at any time.

Operation and timing for various functions are described in the following sections. Parallel binary I/O is not shown since it is assumed that this is implemented with parallel-access static registers which are strobed in every memory cycle.

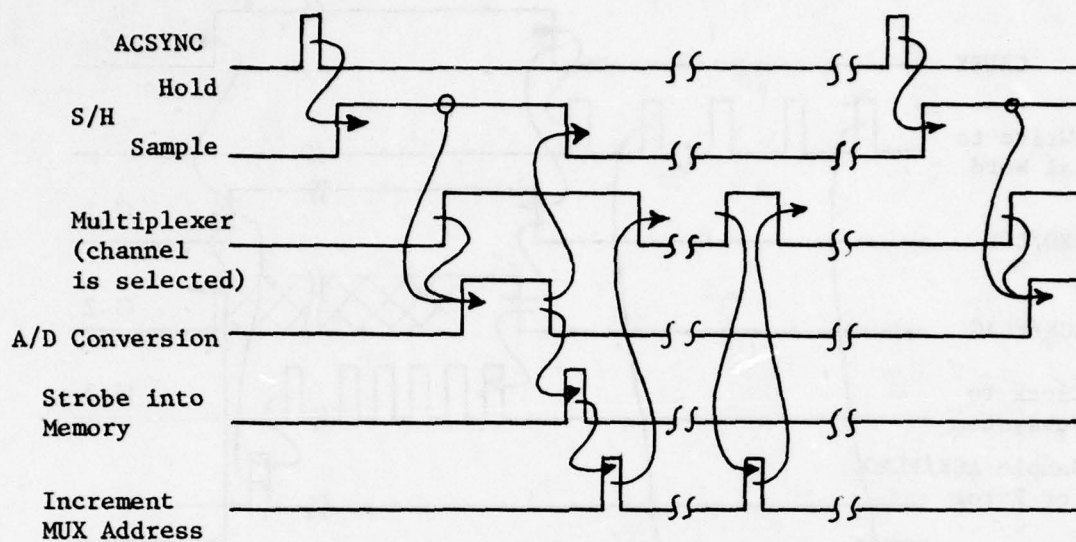
4.3.1 ADC Operation

The ADC of the ICA sequentially samples (via the associated analog multiplexer) the analog signals from all channels and places

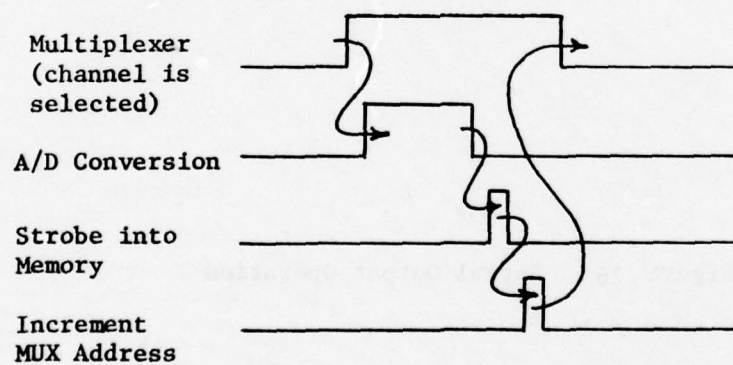
the converted values in the proper analog input words (INAN) of the ICA/LM interface. All channels, except those in the AC mode, are scanned asynchronously at the maximum rate of the ADC, the channel sample-and-hold circuit (S/H) being held in the SAMPLE state. For channels in the AC mode, the signals are sampled by the channel S/H at the time of signal peak as determined by the AC synchronizing signal (ACSYNC), and held there until the next time of ADC conversion for this channel. Once this conversion is complete, no more conversions are made for this channel until after the next AC synchronizing pulse. Timing diagrams for these operations appear in Figure 25 .

4.3.2 Serial Output Operation

The timing diagram for the serial output operation of a channel is shown in Figure 26 . When the channel is not busy (as indicated by the state of the CBUSY status flag), the LMP can write into the serial output word of the LMP interface. When the channel FIFO register becomes full (i.e., the word count reaches the number specified during configuration of the group) the CBUSY flag is set and the channel begins the serial output operation to the subsystem by raising the request line (REQ/LOK). After the subsystem responds by raising the acknowledge line (FLAG/ACK), the ICA transmits the bit stream (including parity bits for each word) and the clock to the subsystem. Upon completion of this transmission, the ICA tests the state of the FLAG/ACK to determine if an error has occurred and lowers the REQ/LOK line. The error condition determined from the FLAG/ACK state is used to clear (no error) or set (error) the ERROR status bit for the channel. Finally it resets the CBUSY flag.



(a) a-c channel



(b) d-c channel

Figure 25 Timing Diagram for Conversion of Analog Inputs

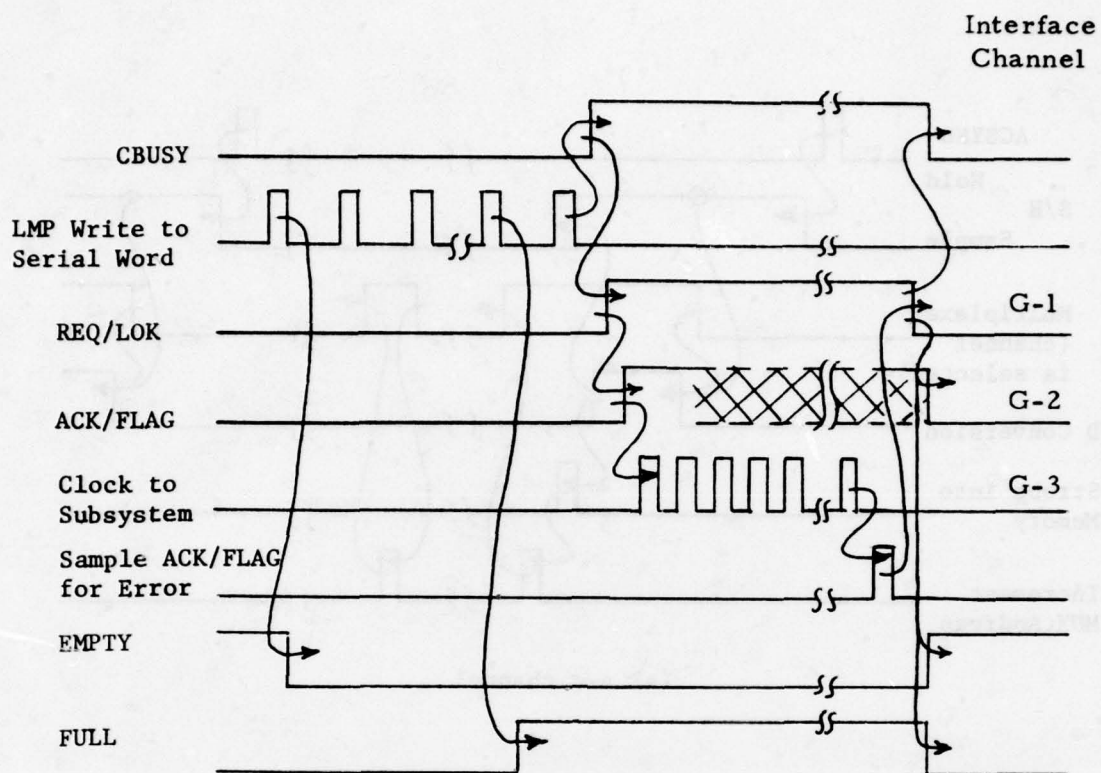


Figure 26 Serial Output Operation

Two status flags provided for programmer convenience are EMPTY, which is set when the FIFO register is empty, and FULL which is set when the next word will fill the FIFO register. The subsystem is expected to lower FLAG/ACK when REQ/LOK is lowered.

4.3.3 Serial Input Operation - Refresh Mode

Refresh timing of serial input operations is controlled by the LMP, which initiates a serial operation by setting the ARM control flag. If the REFRESH configuration flag is set, the ICA begins the serial operation immediately by setting ERROR = 0, clearing the channel FIFO register, and raising the request line (REQ/LOK) to the subsystem. The subsystem responds by raising the acknowledge line (ACK/FLAG) when it is ready to transmit data. The ICA then sets CBUSY = 1, then sends clock pulses to the subsystem causing the system to transmit the data stream including parity bits back to the ICA. The ERROR status bit is set whenever a parity error is detected. When the number of clock bits sent to the subsystem equals 17 times the word count, the ICA terminates the transmission by lowering the REQ/LOK line and clearing the CBUSY and ARM bits. Two status flags provided for programmer convenience are FULL, which indicates that the FIFO register is full, and EMPTY, which indicates that the FIFO register is empty. These flags have no significance while the channel is busy (CBUSY = 1). The timing diagram for this operation is shown in Figure 27 (a).

4.3.4 Serial Input Operations - Flag Mode

Flag mode operations are similar to refresh mode operations

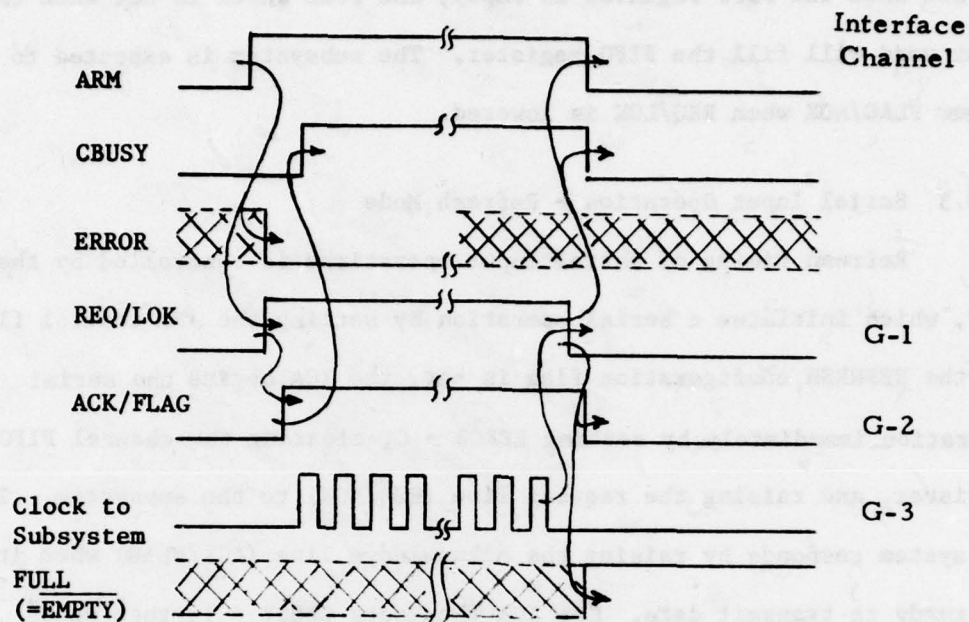


Figure 27 (a) Serial Input Operation - Refresh Mode

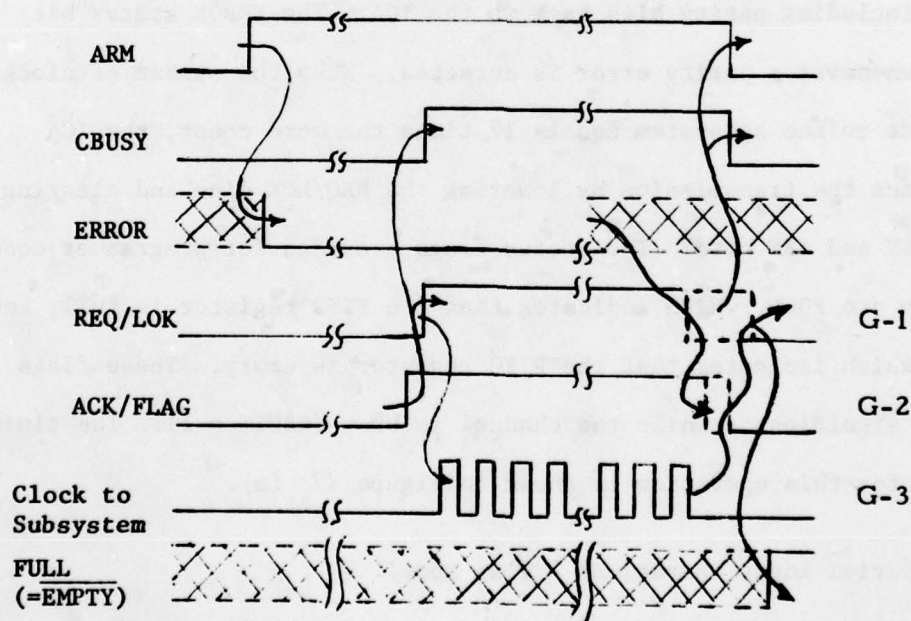


Figure 27 (b) Serial Input Operation - Flag Mode

except that the operation is initiated by the subsystem. The LMP retains control of the operation via the ARM control bit, which must be set before the ICA will recognize the subsystem flag.

Thus, operation begins with the setting of ARM by the LMP. The ICA waits until the subsystem raises the ACK/FLAG line, then it sets CBUSY = 1, raises the channel's REQ/LOK line and begins sending clock pulses to the subsystem. If a parity error is detected, the ICA will lower the REQ/LOK line causing the subsystem to lockout the update of its data and to reset the ACK/FLAG line. After resetting its internal logic the subsystem will retry sending the message. The timing diagram for this operation is shown in Figure 27 (b).

4.4 TESTING

It has already been mentioned that testability is an inherent feature of the ICA architecture. This section presents some details of testing procedures. We shall begin at the subsystem and work inward toward the LMP interface.

Testing of a subsystem depends, of course, on the subsystem design, but some signals, such as serial digital signals, already carry information (parity bits) which inherently provide some testing. The nature of some signals (e.g., analog inputs) is such that subsystem testing requires alternative signal paths and substantial subsystem circuitry. For other types of signals, simple circuit arrangements can be used to provide some information on the state of the connection to the subsystem.

4.4.1 Signal Interface Testing

The ICA signal interface circuits (Figure 22) are tested by

routing the output through the input circuit. For each of the tests, the switch settings are as follows unless otherwise indicated:

- S1 OUTPUT/TEST,
- S2 SINGLE-ENDED,
- S3 OUT,
- S4 DC REFERENCE,
- S5 TEST.

4.4.1.1 Signal Interface TEST 1

The analog output level (ANALOUT) is converted to analog form by the MDAC, and routed through one output filter, the instrumentation amplifier, and the input filter to the sample-and-hold circuit, from which it is read by the ADC of the ICA. Frequency response of the filters can be tested at by suitably varying the output signal. The MDAC input can also be obtained from the LOLEVL and HILEVL inputs, these alternate inputs provide comparison testing of the digital circuits in preceeding the MDAC.

4.4.1.2 Signal Interface TEST 2

Switch S2 is set to position DIFFERENTIAL. The output signal is routed through the inverting amplifier thereby testing this device and the common-mode rejection characteristic of the instrumentation amplifier.

4.4.1.3 Signal Interface Test 3

The group threshold (TRSHLD) is set to various levels to test

the comparator.

4.4.1.4 Signal Interface TEST 4

Switch S4 is set to position AC REFERENCE. This test is for comparison of the levels of the two references (AC and DC).

4.4.1.5 Signal Interface TEST 5

Switch S1 is set to position INPUT, and switch S3 is set to position ON. Current from one source is routed through the test impedance (R_{TEST}), the voltage across which is measured by the channel input circuit. This tests the current source.

4.4.1.6 Signal Interface TEST 6

Switch S1 is set to position INPUT, switch S3 is set to position ON, and switch S2 is set to DIFFERENTIAL. This is the same as TEST 5, except that the current sources are tested together (in series as they are used in differential operation).

4.4.2 ADC Testing

The tests above depend on the integrity of the ADC and associated analog multiplexer of the ICA. Since the testing process routes signals from each of the 16 MDAC's to the ADC and compares the ADC output with the MDAC inputs, this procedure thoroughly tests the ADC and the analog multiplexer. It is assumed, of course, that the reference level for the ADC is derived independently (preferably internally to the ADC) from the DC and AC reference levels for the MDAC's.

4.4.3 Serial I/O Testing

Serial output signals are looped back through the serial input

hardware, and the output data and error condition (obtained from the output parity error bits) are stored in appropriate input registers of the channel. Thus, the LMP has access to complete information on the state of the output. Channels used for serial input can be tested in a similar way in the TEST state, in which the subsystems are electrically disconnected from the ICA.

Note, that correctness of the parity bit in either situation does not guarantee that the parity error detection circuitry is functioning properly, since bit streams looped back in this manner would normally have correct parity. The circuitry is tested by performing the test operation with the BADPAR control bit set. In the TEST state, BADPAR causes the bit stream to be generated with parity inverted, hence always incorrect.

Testing of subsystem parity detection should not be implemented in the same way, since improper operation might cause unwanted subsystem actions. Thus, BADPAR cannot be set in the RUN state.

SECTION V

SUBSYSTEM INFORMATION CHANNEL

The Subsystem Information Channel (SIC) is the channel over which the Link Module communicates with a subsystem's electronic nameplate (NP). The electronic nameplate is a memory device attached to each subsystem. It contains information about the subsystem which the LM must have in order to properly interface to the subsystem. This information includes: the subsystem's EID, the nameplate configuration (described below), interface configuration details for the LM's ICA, data conversion routines which the LM will apply to data received from or sent to the subsystem, diagnostic test routines to exercise the subsystem to verify proper operation, the calibration record of the subsystem, and a log of errors which occur during normal operation of the subsystem.

During system initialization, after establishing communications with the nameplate, the LM will first check the EID and nameplate configuration to confirm that the proper subsystem and all its components are present. The interface configuration information is then obtained from the nameplate and is used to configure the LM's universal interface to properly mate with the subsystem. Then the data conversion routines are extracted to be used in processing all data transferred to and from the subsystem.

Also, either during initialization or during normal operations, the LM can extract the diagnostic test routines and execute them to test the proper operation of the subsystem. The calibration record is available for the LM to verify that the subsystem installed has received any required calibration. The calibration record includes offset parameters which can be used by the data conversion routines to correct values obtained from or sent to the subsystem.

Figure 28 illustrates a typical nameplate configuration. When several small subsystems are connected together as a unit and interface via a single Link Module, the individual electronic nameplates are chained together on the LM's single SIC. In some situations an additional master nameplate might be applied to the entire integrated unit, with that nameplate the first in the chain. To allow for this sort of integration, all nameplates include a "nameplate configuration" which lists which, if any, nameplates must be chained to it in order to have a proper equipment configuration. Section 5.2 describes the SIC commands which facilitate communication with each of the nameplates.

A master nameplate not only tells which additional nameplates should be present in the system, but also tells how signal lines on the universal interface are allocated among the bundled subsystems. For example, as illustrated in Figure 29 suppose subsystem #1's nameplate states that 4 analog input lines are needed, and subsystem #2's nameplate states that 8 analog output lines are needed. When the two subsystems are integrated into a single system with signal lines bundled in one connector and cable to the LM, a determination must be made regarding which of the LM's 16 signal lines are allocated to which subsystem. The master nameplate tells the LM how that allocation was made, indicating, for example, that lines 1-4 of the LM are to be configured for subsystem #1, and lines 5-12 of the LM are to be configured for subsystem #2. In the absence of a master nameplate providing this information, the LM assumes by default that lines are allocated in order according to where the individual nameplates appear in the chain.

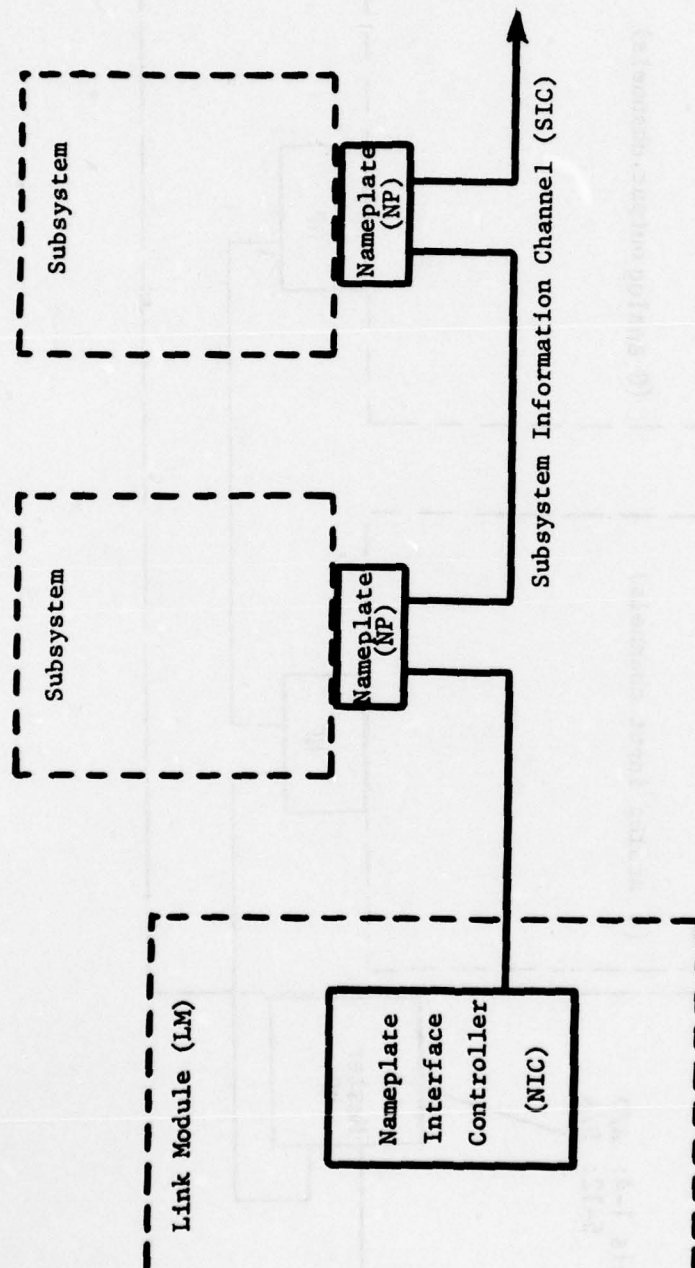


Figure 28 Communication With Nameplate through the Subsystem Information Channel

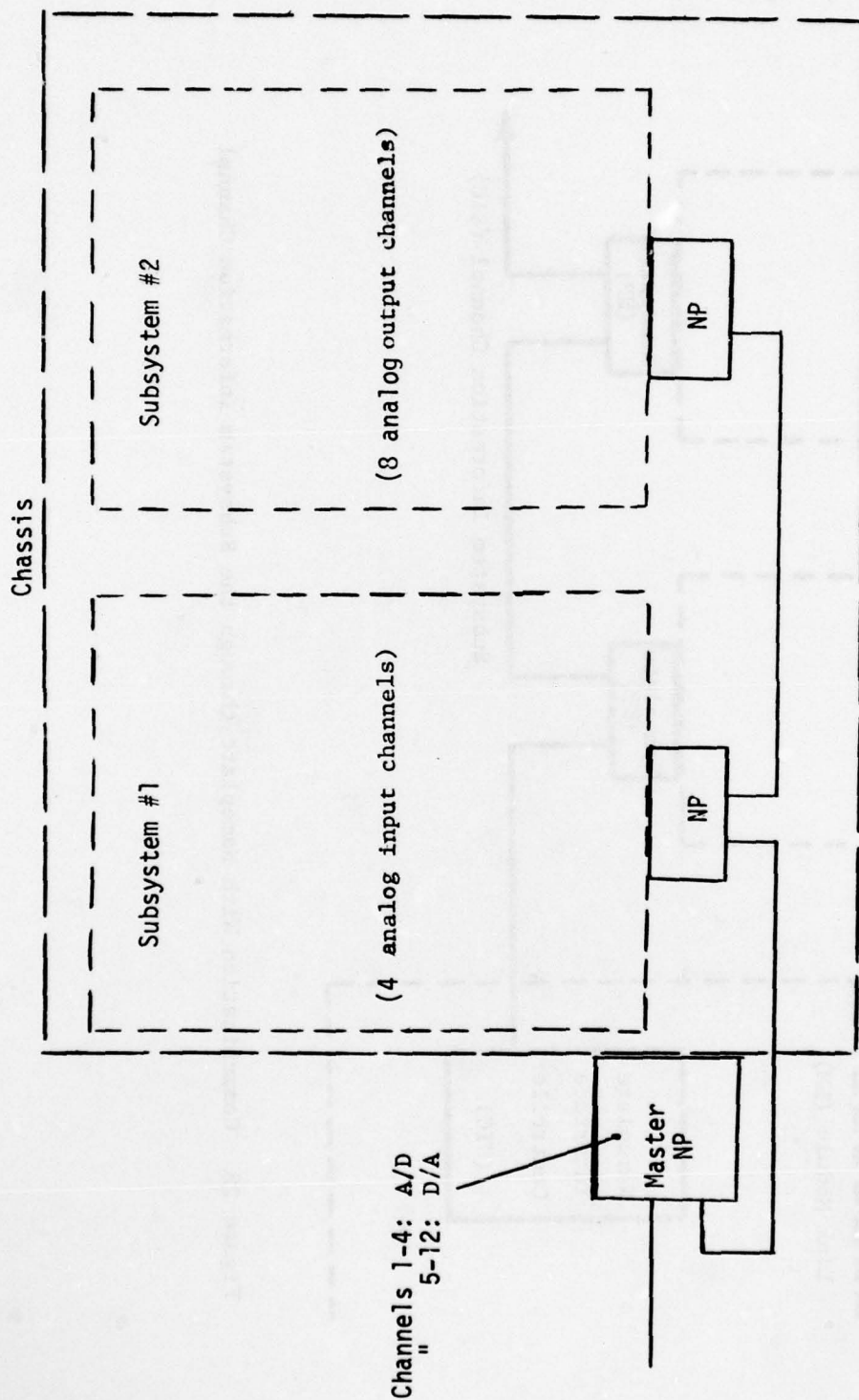


Figure 29 Master Nameplate Configuration

5.1 COMMUNICATION PROTOCOL

Figure 50 illustrates the format of the serial communications on the subsystem information channel. A single fixed-length command/data word is transmitted by the Link Module, and a single fixed-length status/data word is always returned by the selected nameplate.

The command field, status field, and data field are each 8 bits long. Thus the command/data word totals 19 bits: a start bit, 16 bits for command and data, a parity bit, and a stop bit. The status/data word is similarly 19 bits long. For those commands for which the data field is not utilized, the field is arbitrary and is ignored.

A clock signal is transmitted to the nameplates by the LM. The clock is held low between commands. It is restarted when a command is transmitted and is maintained until after the status word is received. It must then return to zero for a minimum period, called the command gap, before a new command can commence. This procedure provides a way of resetting all nameplates by holding the clock low. It also allows nameplates to distinguish command words from status words on the bus.

There is also a status gap between the command and status words. After receipt of a command word, the nameplate must return status before the end of the status gap. A delay longer than this indicates to the LM that the command was not received. The LM would then retry the command.

5.2 COMMANDS AND STATUS

Only one nameplate is "selected" for communication at a time. Selection is performed by one of two methods: either by nameplate

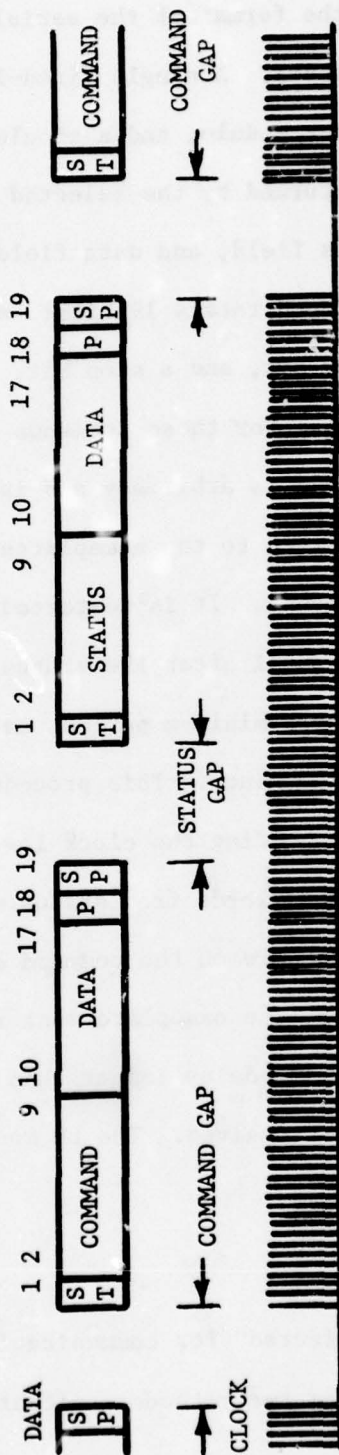


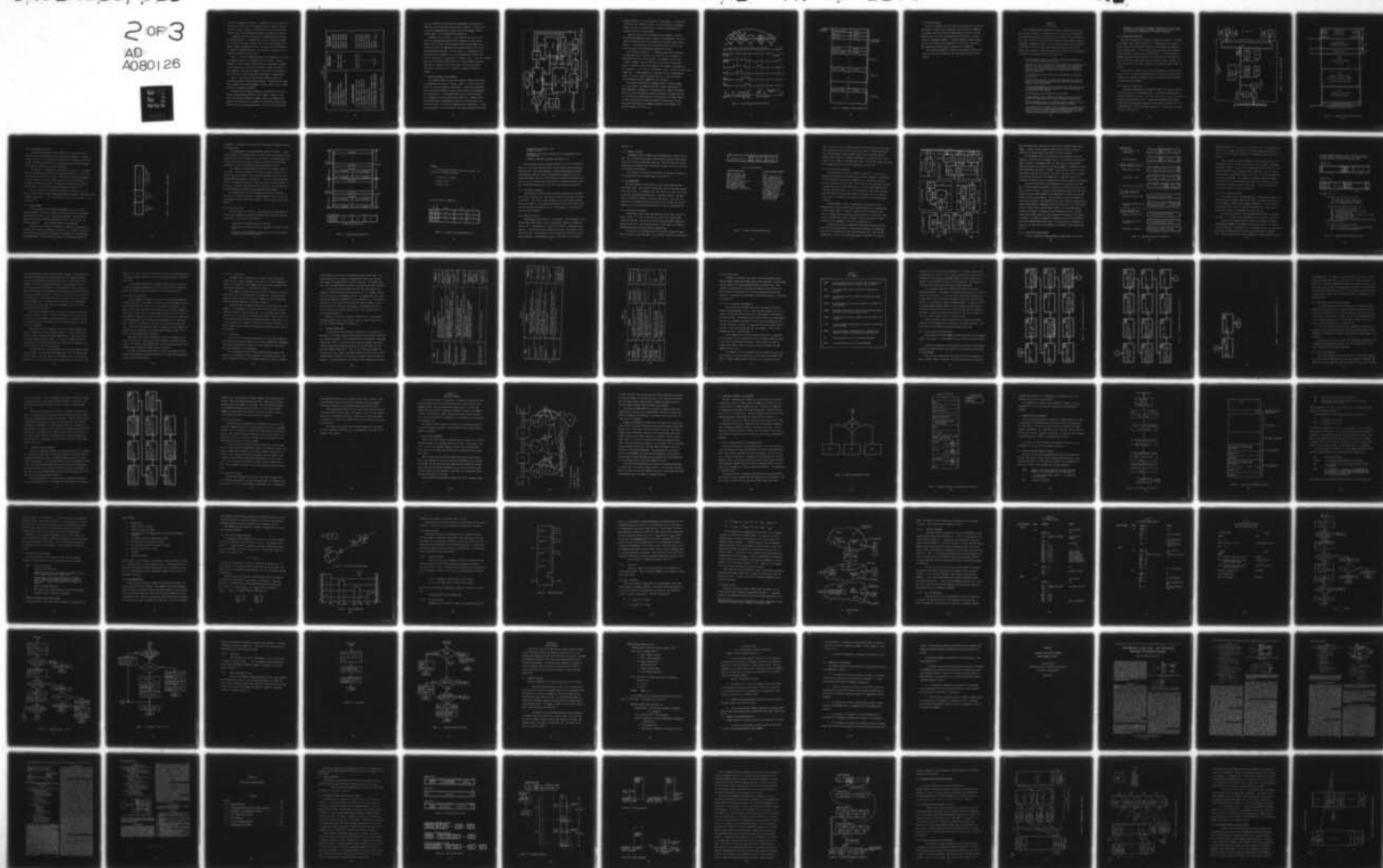
Figure 30 Message Format on the Subsystem Information Channel (SIC)

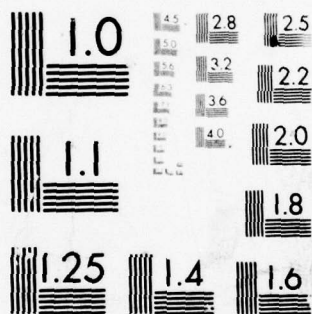
AD-A080 126

HOUSTON UNIV TX DEPT OF ELECTRICAL ENGINEERING
REMOTE LINK UNIT FUNCTIONAL DESIGN: AN ADVANCED REMOTE TERMINAL--ETC(U)
OCT 79 C J TAVORA, J R GLOVER, G W BATTEN F33615-78-C-1634
AFAL -TR-79-1176 NL

UNCLASSIFIED

2 OF 3
AD
A080126





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

"level" or by nameplate "address". A nameplate's level is indicated by its position in the chain. The nameplate nearest the LM is level 0, the next is level 1, and so on. The priority line in the SIC and the priority logic on each nameplate facilitate this method of selection.

A nameplate can also be selected for communication by selecting its nameplate address. A nameplate must first be selected according to its level in the chain, at which time it can be initialized with a specific address assigned by the LM. From then on it can be selected at random for communication by way of its address, without the need of stepping through the priority chain.

Table 9 lists the commands proposed for the nameplate. The first six commands pertain to the nameplate selection process. Normally commands 1, 2, and 3 would be used during an initialization procedure to step through the priority chain assigning nameplate addresses to each nameplate. From then on commands 4 and 5 would be the commands most often used to select and deselect nameplates for communication.

Table 9 also lists the data transfer commands proposed for accessing the memory of the selected nameplate. No distinction is made in the commands between ROM and EAROM. ROM will simply occupy a different block of addresses from EAROM.

The status word is always returned by the nameplate which is the selected nameplate after execution of the command. However, in the case of the "deselect nameplate" command, there is no nameplate left selected; the status word is returned by the previously selected nameplate, the one which is being deselected. One bit of the status word

TABLE 9
NAMEPLATE COMMANDS

<u>NAMEPLATE SELECT COMMANDS</u>	<u>DATA FIELD IN COMMAND WORD</u>	<u>DATA FIELD IN STATUS WORD</u>
1. Select level 0	X	nameplate address
2. Select next level	X	nameplate address
3. Assign nameplate address	nameplate address	nameplate address
4. Select nameplate by address	nameplate address	nameplate address
5. Deselect nameplate	X	nameplate address
6. Read selected nameplate address/status	X	nameplate address
<u>DATA TRANSFER COMMANDS</u>		
7. Set memory pointer, high byte	pointer high byte	pointer high byte
8. Set memory pointer, low byte	pointer low byte	pointer low byte
9. Read memory pointer, high byte	X	pointer high byte
10. Read memory pointer, low byte	X	pointer low byte
11. Write/erase enable	X	nameplate address
12. Write/erase disable	X	nameplate address
13. Read memory data, increment pointer	X	data byte
14. Write memory data, increment pointer	data byte	data byte
15. Erase memory data, increment pointer	X	nameplate address
X = doesn't matter, field is ignored		

is set to indicate that the nameplate acknowledges being deselected. Other bits in the status word will be used to indicate: a parity error in the last command/data word received, an invalid command received, and an attempt to write without "write enable" set.

Commands 1 and 2 in Table 9 are special in that they have the effect of simultaneously deselecting one nameplate and selecting another. Only the newly selected nameplate should return status, but a reception error by the nameplate being deselected might result in two status words being returned. For this reason, if a nameplate is ever selected by way of command 1 or 2, it will automatically deselect and return no status upon a reception error. Commands 1 and 2, along with command 3, will only be used during initialization to assign nameplate addresses. From then on, commands 4 and 5 will be used to select and deselect nameplates, and status will be returned by a nameplate selected by command.

5.3 ELECTRONIC NAMEPLATE ARCHITECTURE

Each nameplate appears to the Link Module as a memory device which is communicated with by way of the SIC. Figure 31 shows a block diagram of the proposed nameplate architecture. A Universal Asynchronous Receiver/Transmitter (UART) or similar device receives the command/data word and latches it. The command decoder and controller determines which command was received and initiates execution. A memory pointer sequentially moves through memory as words are read and written, but can also be reset to any value by command from the Link Module. A clock detector/generator circuit receives a clock signal generated by the Link

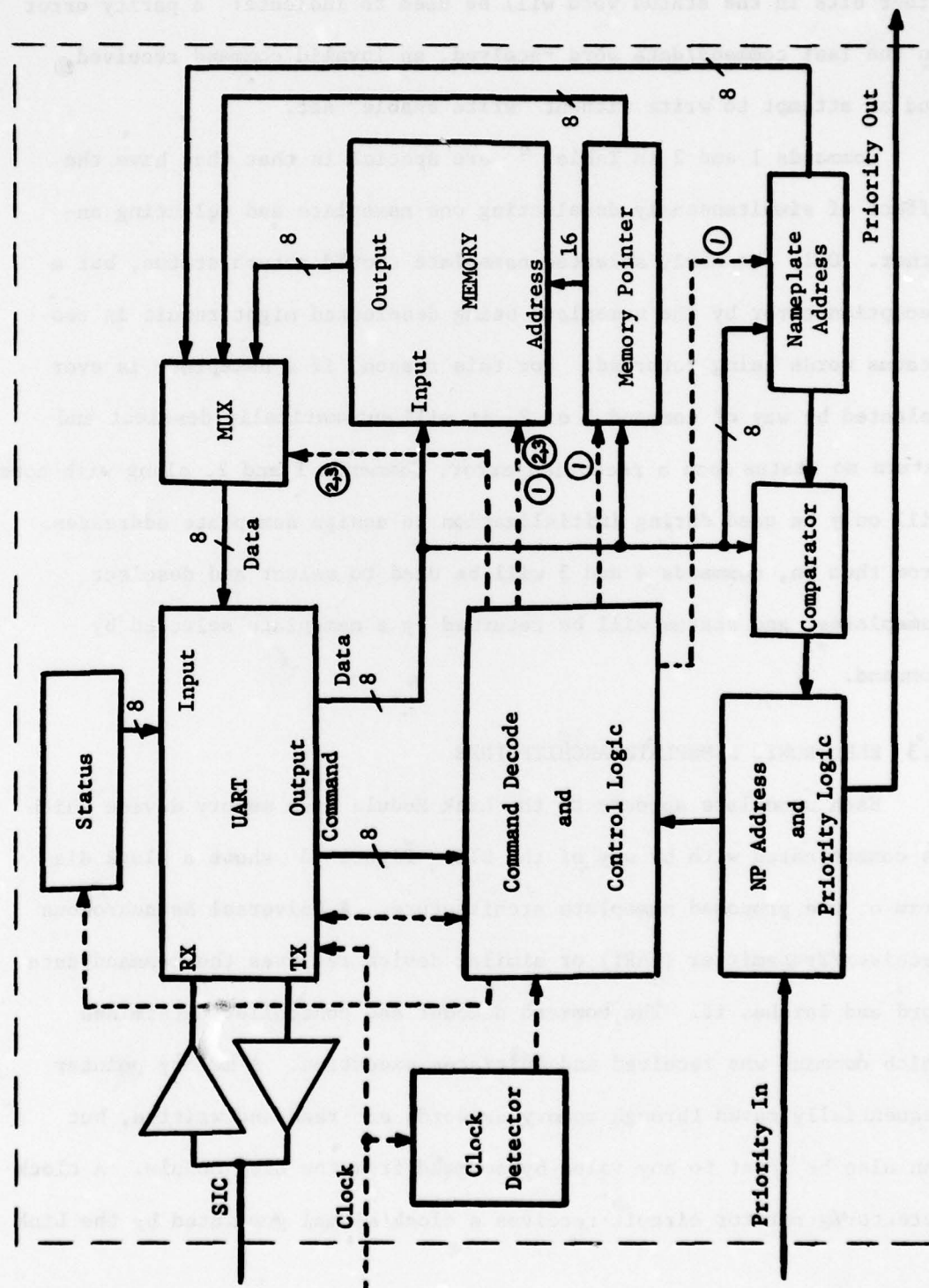


Figure 31 Nameplate Architecture

Module and uses it to clock operations in the nameplate. The UART also transmits status and data to the LM. The blocks relating to nameplate address are used to determine which of several chained nameplates is presently selected for communication.

Figure 32 shows the state diagram for the nameplate controller and the timing diagram for control signals on the nameplate. The state numbers correspond to the circled numbers on the control lines shown in Figure 31, indicating the time period during which the control lines would be active if the appropriate command were received.

As illustrated in Figure 30, a command sequence begins following a command gap period during which the LM holds the clock signal low. Upon detection of the clock once again, the nameplate enters state 1 to await a command. When the UART's data ready (DR) line goes high, state 2 is entered to execute the command if the command is addressed to this nameplate. If not, there is an immediate return to state 0 to await a new command gap. If this nameplate is to respond, it will proceed through states 2, 3, and 4, before returning to state 0. State 3 performs a read from memory only for commands 13 and 14 in Table 9.

The proposed organization of nameplate memory is shown in Figure 33. All entries in the memory begin with a header consisting of an entry identifier, followed by a word indicating the length of the entry. Entry 0 is always the nameplate directory and begins in memory location 0. The directory contains pointers to the headers of all other entries. The other entries would include the nameplate configuration, the subsystem EID, data conversion programs, diagnostic test programs, calibration records, error logs, etc.

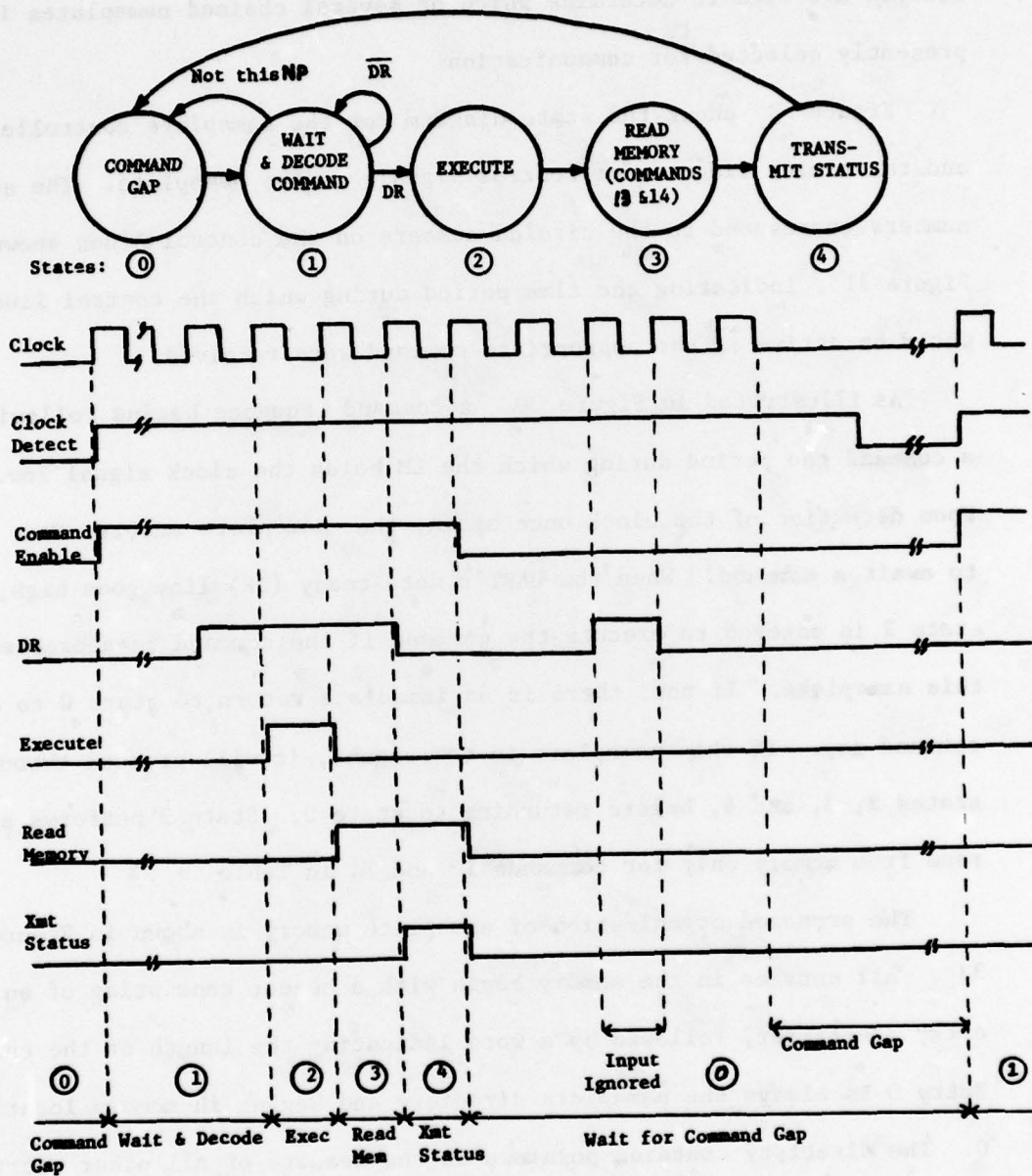


Figure 32 State Diagram and Timing Diagram

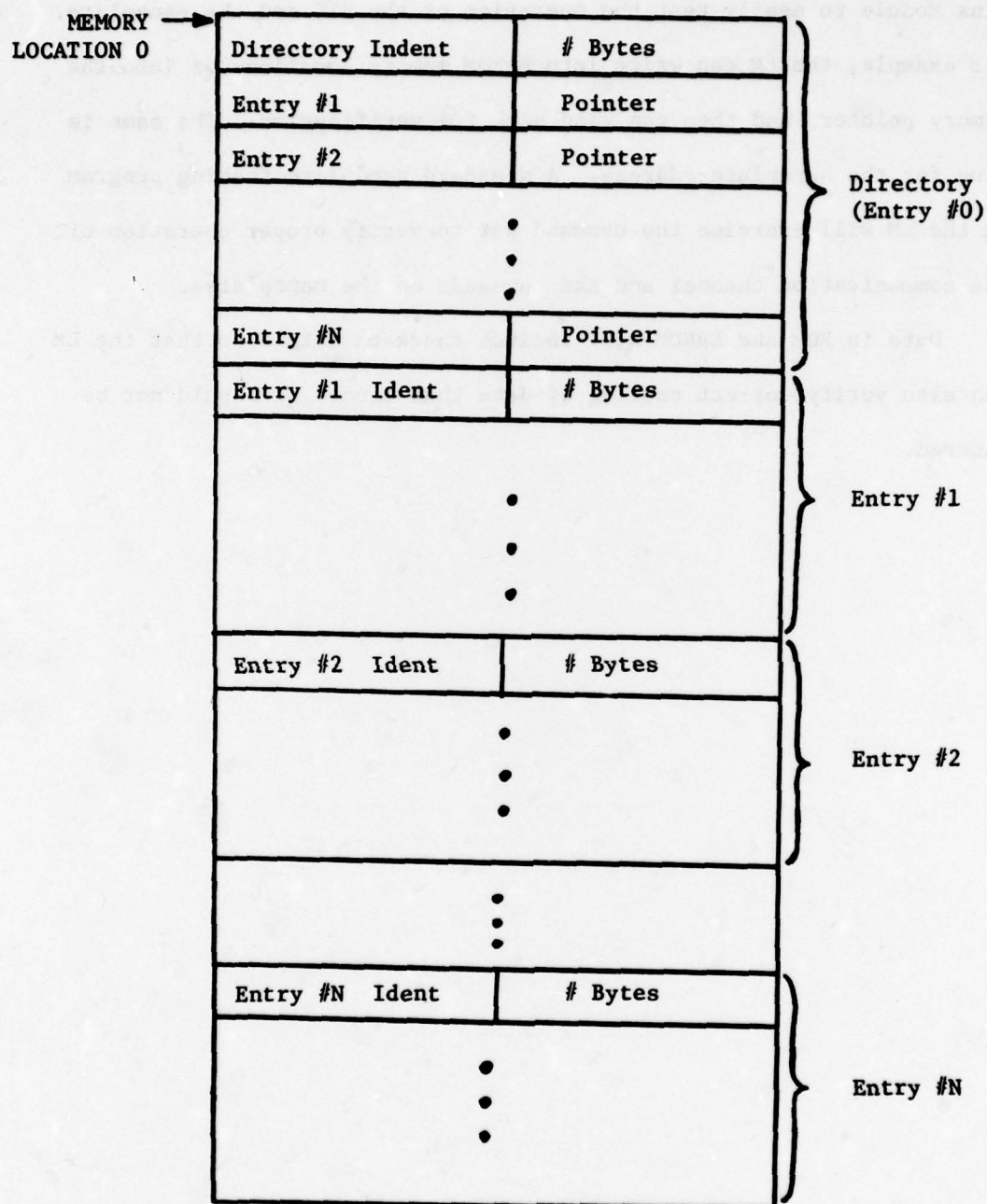


Figure 33 Nameplate Memory Organization

5.4 TESTING PROCEDURE

The set of commands and status codes have been chosen to allow the Link Module to easily test the operation of the SIC and the nameplate. For example, the LM can write into EAROM memory locations or into the memory pointer, and then can read back for verification. The same is true for the nameplate address. A standard nameplate testing program in the LM will exercise the command set to verify proper operation of the communication channel and the commands on the nameplates.

Data in ROM and EAROM will include checksum totals so that the LM can also verify correct reading of data that cannot or should not be altered.

SECTION VI

LINK MANAGER

The Link Manager (LMG) supervises the overall operation of the RLU and each of the Link Modules. Responsibilities of the LMG are divided between two intelligent processors, the Manager Processor (MGP) and the Link Control Unit (LCU). The MGP is responsible for execution of mode commands, failure management, maintenance support, and has ultimate authority for the operation of the LMG. The LCU is a high-speed, semi-intelligent controller which transfers data between the Multiplex Terminal Unit (MTU) and the Link Modules. The LMG has the following features and capabilities:

- The LMG utilizes the present DAIS MTU to provide the interface between the Mil-Std-1553B bus and the LCU.
- The LCU is a high-speed controller which implements all communications with the MTU. It is essentially a direct memory access (DMA) controller on the LMG's internal bus which transfers data between the MTU and either the MGP's random access memory (RAM) or the shared memory with the LM's.
- The LCU maintains the bus status word and implements all multiplex bus protocols.
- The LCU notifies the MGP of all mode code operations which must be executed, and automatically performs the single data word transfers required by mode codes 16-31.
- The Last Command Register, BIT word, and activity register are stored in dedicated MGP RAM locations accessible to the LCU in response to the appropriate mode commands.
- The MGP executes mode code operations as required.
- The MGP maintains tasks capable of providing local control of subsystems if communications with the DAIS processors should lapse.
- Failure management tasks in the MGP bring back-up subsystems on line when the primary subsystems are reported down by the LM's.
- A maintenance port to the MGP allows all functions of the LMG and the LM's to be directed or monitored by maintenance personnel or a maintenance computer, including simulation of communications over the multiplex bus without using either the MTU or the LCU.

- Subaddress 1 is allocated to messages communicated directly between the MGP and the DAIS master processor, allowing full control of all RLU functions and capabilities by the master executive.

6.1 LINK MANAGER ARCHITECTURE

The basic architecture of the LMG is diagrammed in Figure 34 . The MGP is bus master over the LMG's internal bus. A memory-mapped I/O scheme is used so that all other components on the bus occupy segments of the MGP's physical address space. A 16-bit address word is used with byte addressing, allowing 64K bytes of memory and I/O address space. A 16-bit data word is assumed, though both words and bytes can be transferred on the internal bus. Bus control signals include read, write, byte/word, LCU/MGP, interrupt controls, DMA controls, and others required by the specific processor chosen for the MGP.

The allocation of address space is shown in Figure 35 and is described further below. Allocations indicated there and the specific addresses used throughout this design are examples that may require modification depending on the MGP processor selected.

6.1.1 Multiplex Terminal Unit

The present DAIS Multiplex Terminal Unit (MTU) is retained in the RLU for interfacing to the 1553B multiplex bus. The interface signals now presented to the DAIS RT's Timing and Control Unit (TCU) are instead presented to the RLU's Link Control Unit (LCU). An MTU Control Interface (MCI) allows the MTU mode code operations (power on/off, Bus A/Bus B selection, etc.) to be implemented by the MGP via the LMG internal bus. A description of the MCI is given in Section 6.1.5.

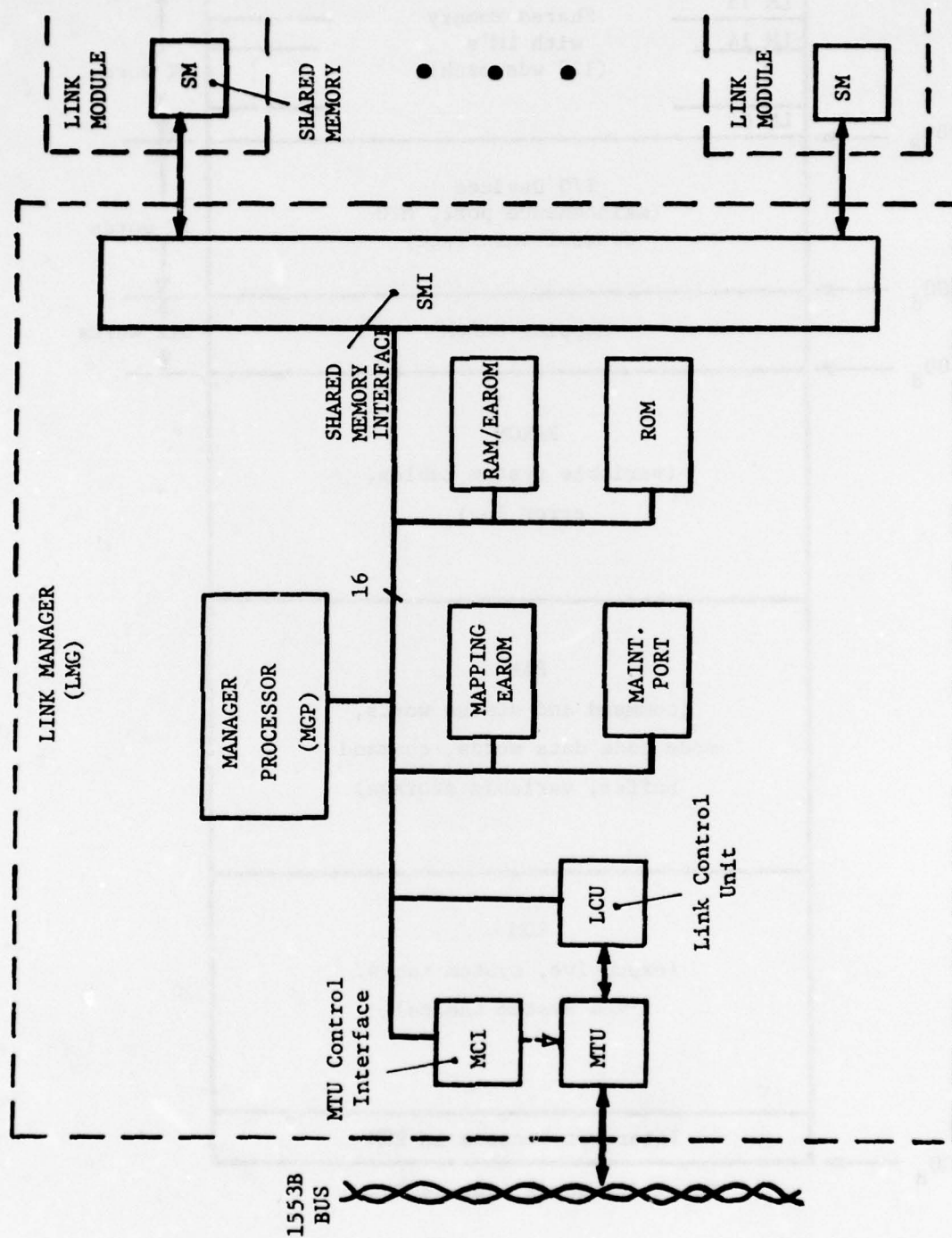


Figure 34 Architecture of Link Manager

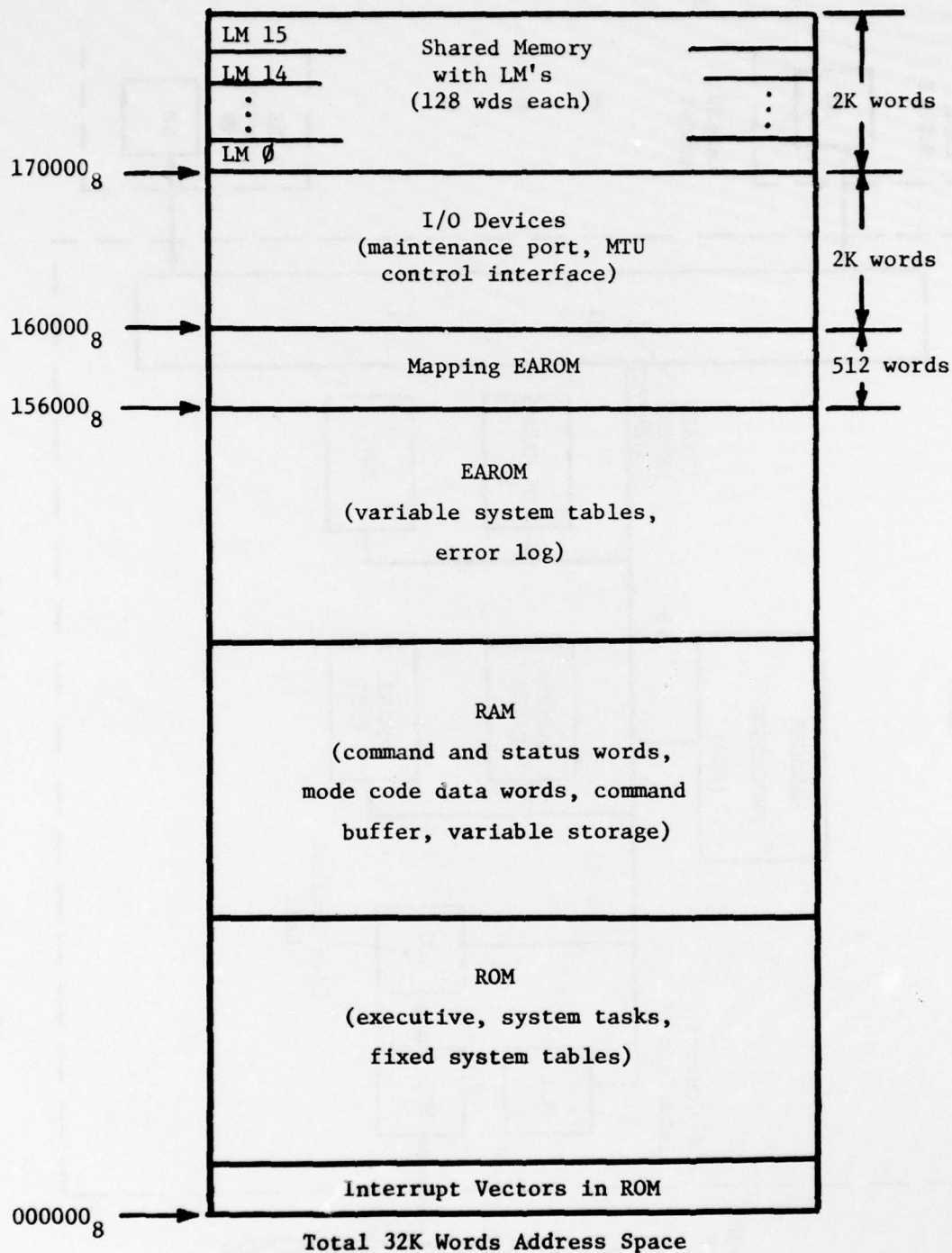


Figure 35 LMG Memory and I/O Address Space

6.1.2 Shared Memory Interface

The LMG's Shared Memory Interface (SMI) maps all accesses to the LMG's 2K words of shared memory to the appropriate Link Module. As illustrated in Figure 35, the 2K words are divided into 16 sections of 128 words per LM. The 128 words allocated to each LM represent that LM's SM interface as described in Section 3.1.1 on the Link Module.

The mapping of the addresses is illustrated in Figure 36. The LMG's SMI is selected when bits 15-12 of the address word are all 1's. Bits 11-8 are then used to select one of 16 LM's. Only the low 8 bits of the address are then presented to the LM, allowing access to 256 bytes (128 words) of virtual shared memory in the LM. The mapping of these 8 bits to each LM's physical memory is more involved and is described in Section 3.1.1.

The very last byte in each LM's shared memory is special in that the SMI will map all writes to one of these locations to all 16 LM's. This byte is Data Transfer Command byte C1. For more information on its use, see Section 3.1.3 and Section 6.2.

6.1.3 Mapping EAROM

A 512-word Electrically Alterable ROM (EAROM) is used by the LMG to map the subaddress (SA) in the command received from DAIS to the Link Module and Link Address (LM/LA) of the corresponding subsystem. This mapping is similar to the 512-byte EPROM now employed in the RT to map the SA to the Interface Module and Interface Channel (IM/IC) of the subsystem.

During RLU initialization, the MGP obtains subsystem configuration information from each LM, as described in Section 3.2.2. The MGP also receives from the DAIS master executive the subaddress assignments for each subsystem. The mapping EAROM is then loaded with a table of correspondence between SA

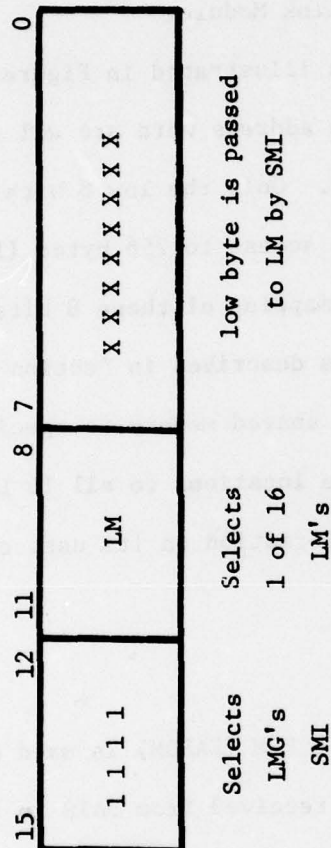


Figure 36 Address Word Format for Shared Memory

and LM/LA. A description of the entire LMG initialization sequence is given in Section 6.3.2.

The organization of the mapping EAROM is shown in Figure 37 , along with the format of each 16-bit entry. For each SA there are pointers to two message lists, one for transmit and one for receive. Each message list consists of one or more map entries. There is one map entry for each LM/LA combination to which a portion of the entire message is directed.

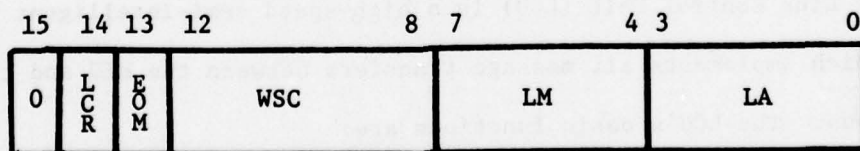
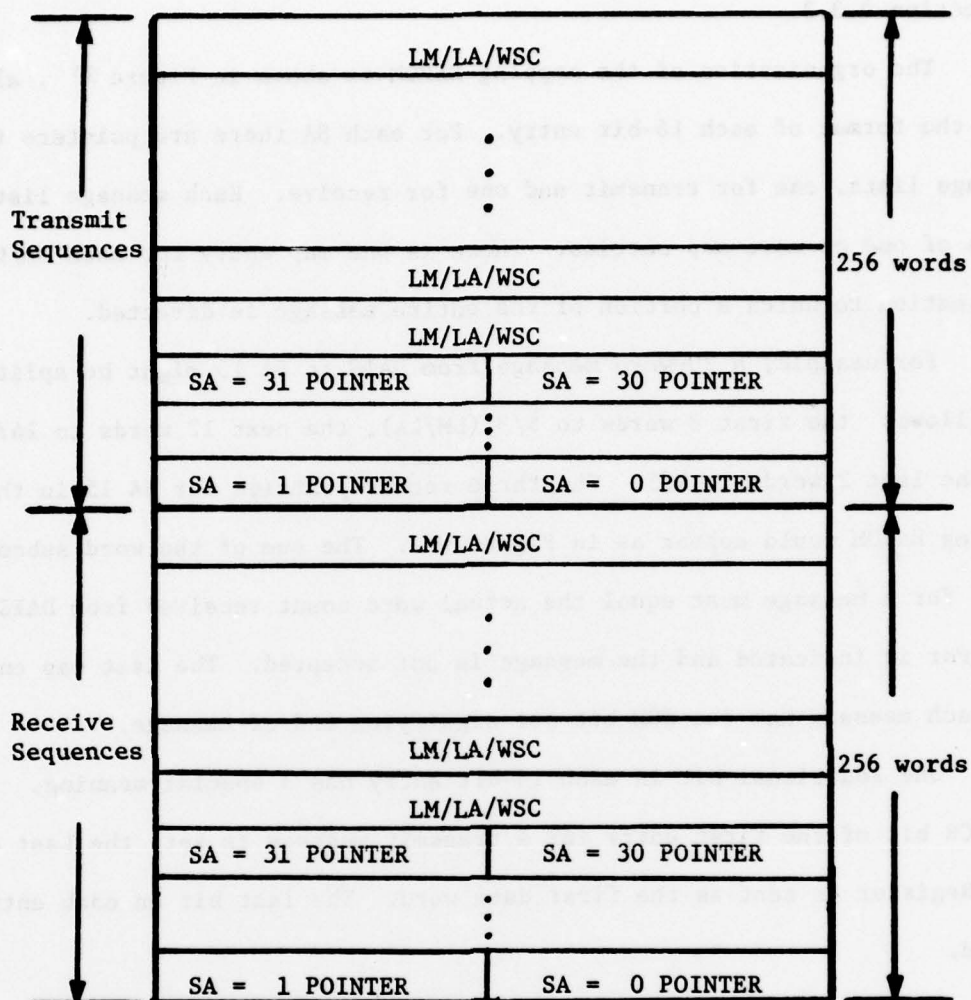
For example, a 20-word message from DAIS to SA 15 might be split up as follows: the first 8 words to 5/3 (LM/LA), the next 12 words to 14/4, and the last 2 words to 9/5. The three receive entries for SA 15 in the mapping EAROM would appear as in Figure 38 . The sum of the word subcounts (WSC) for a message must equal the actual word count received from DAIS, or an error is indicated and the message is not accepted. The last map entry for each message has the EOM bit set signifying end-of-message.

One additional bit in each 16-bit entry has a special meaning. If the LCR bit of the first entry for a transmit message is set, the Last Command Register is sent as the first data word. The last bit on each entry is unused.

6.1.4 Link Control Unit

The Link Control Unit (LCU) is a high-speed semi-intelligent DMA controller which implements all message transfers between the MTU and the LMG internal bus. The LCU's basic functions are:

- Disposition of messages received from the MTU.
- Implementation of 1553B protocols on messages for the MTU to transmit.
- Execution of all handshaking control/status with the LM's on data transfers to/from the shared memory.



Format of Each Map Entry

Figure 37 Mapping EAROM Organization

Example

A 22-word receive message from DAIS to some SA. The SA maps to a three-part message:

8 words to 5/3 (LM/LA)

12 words to 14/4

2 words to 9/5

The three entries in EAROM are:

L E C O R M			WSC	LM	LA
0	0	0	8	5	3
0	0	0	12	14	4
0	0	1	2	9	5

Figure 38 Example of Mapping EAROM Entries

- Notification to the MGP of mode commands received.
- Maintenance of the status register and its transmission on the multiplex bus.
- Transfer to MGP RAM of messages addressed to SA 1.

The LCU differs from the present RT Timing and Control Unit (TCU) in that it is a far less complicated device which performs only the functions described above. Mode code operations, terminal and subsystem self-tests, analog-digital conversion, and all handshaking with subsystems for both serial and parallel data are functions which are presently performed by the TCU in an RT, but which have been distributed to the MGP and the LM's in the RLU. The detailed architecture of the LCU is given below in Section 6.2.

6.1.5 MTU Control Interface

The MTU Control Interface (MCI) provides the means by which the MGP and the LCU can execute mode code operations which directly affect the state of the MTU's. It is simply a memory-mapped I/O port which takes a control word from the LMG bus. By setting the appropriate bits in the control word, the MGP can enable and disable each MTU. This capability is necessary for some mode code operations.

6.1.6 Maintenance Port

The Maintenance Port (MP) is a Mil-Std-188 serial interface on the LMG bus which allows either a CRT or a maintenance computer to communicate with the MGP. A maintenance utility task in the MGP will communicate with this port. The task will accept commands which direct the operation of the LMG and the LM's, or which simply allow the operation of the RLU to be monitored externally. The maintenance utility is described in more detail in

Section 6.3.

6.1.7 Manager Processor

The Manager Processor (MGP) is the intelligent host processor for the LMG. It is a 16-bit microprocessor with vectored interrupts, direct memory access, and a 64K-byte addressing capability. There are several new 16-bit microprocessors available which could be used, though a DEC LSI-11 was used as an example in this design.

The functions of the MGP are described by the software it executes. A full description of this software appears in Section 6.3.

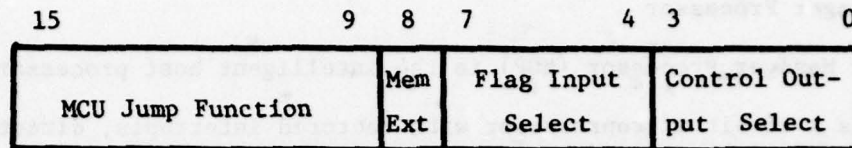
6.2 LCU ARCHITECTURE

The LCU architecture described here uses an Intel 3001 Microprogram Control Unit (MCU) as an example, although some other microprogram sequencer could also be used. An arithmetic logic unit (ALU) is not used with the MCU, as only the conditional sequencing logic of the MCU is needed. The small amount of arithmetic necessary (counting, comparisons) is more easily performed by a few peripheral MSI circuits than by use of an ALU in a full implementation of a bit-sliced microprocessor as is done in the present TCU.

6.2.1 Microinstruction Format

The MCU uses a short 16-bit microinstruction, the format of which is shown in Figure 39. Seven of the bits are used for the standard jump function of the 3001. The next bit is used to extend the MCU's addressing capability from 512 to 1024 microinstructions. These eight bits combine to determine which microinstruction will be executed next.

The next four bits determine which of 16 possible single-bit inputs will be selected as the "flag input" to be tested by conditional jump instruc-



16-bit Microinstruction

Possible Flag Inputs

EOM bit from map entry
 LCR bit from map entry
 Word subcounter = WSC
 Total counter = WC
 RDY bit from status byte S0
 Mode command received
 LMG command received
 LMG bus status signals
 MTU handshake status signals

Possible Control Outputs

Reset
 Write command byte C0
 Write command byte C1
 Read status bytes S0,S1
 Write status bytes S0,S1
 Transfer data on LMG bus
 Transfer MTU data and
 increment counters
 Set status word bits
 Transmit status word
 Read EAROM pointer
 Read EAROM map entry and
 increment pointer
 Read LCR from RAM
 Zero word counter

Figure 39 Format of LCU Microinstruction

tions. The last four bits determine which of 16 possible output control lines will be asserted to perform some function in the LCU. The MCU thus performs as a semi-intelligent control sequencer which tests flag inputs, and then asserts control outputs in a manner determined by the results of these tests. Figure 39 also suggests flag inputs and control outputs which might be selected by a microinstruction in an actual implementation of the LCU.

6.2.2 LCU Functional Components

The architecture of the LCU is shown in Figure 40 . For ease of explanation the diagram is partitioned into five major components: microcontroller, MTU interface, timing logic, arithmetic logic, and LMG bus interface. The microcontroller section is simply the MCU and the logic and memory necessary to encode and decode the microinstructions. The control ROM (CROM) contains the 16-bit microinstructions. The 16-to-1 multiplexer selects the flag input to the MCU, and the 4-to-16 decoder selects the control line to be asserted. A reset line is provided from the LMG bus to the MCU to allow initialization of the LCU.

The MTU interface receives command and data words from the MTU. A command word is latched and the Terminal Address (TA) field is compared with the RLU's address plug. If there is a match, the MCU is notified of the start of a message and of each data word thereafter as it is received. Flag inputs to the MCU are provided to indicate an error detected by the MTU, receipt of a mode command, or receipt of a command for SA 1.

A timing logic section is required to facilitate synchronization between the MTU, LCU, and LMG bus. Delay circuits implement the timing gaps required in the 1553B protocol and on the LMG internal bus. When the MCU commands that a read or write take place either on the LMG bus or with the

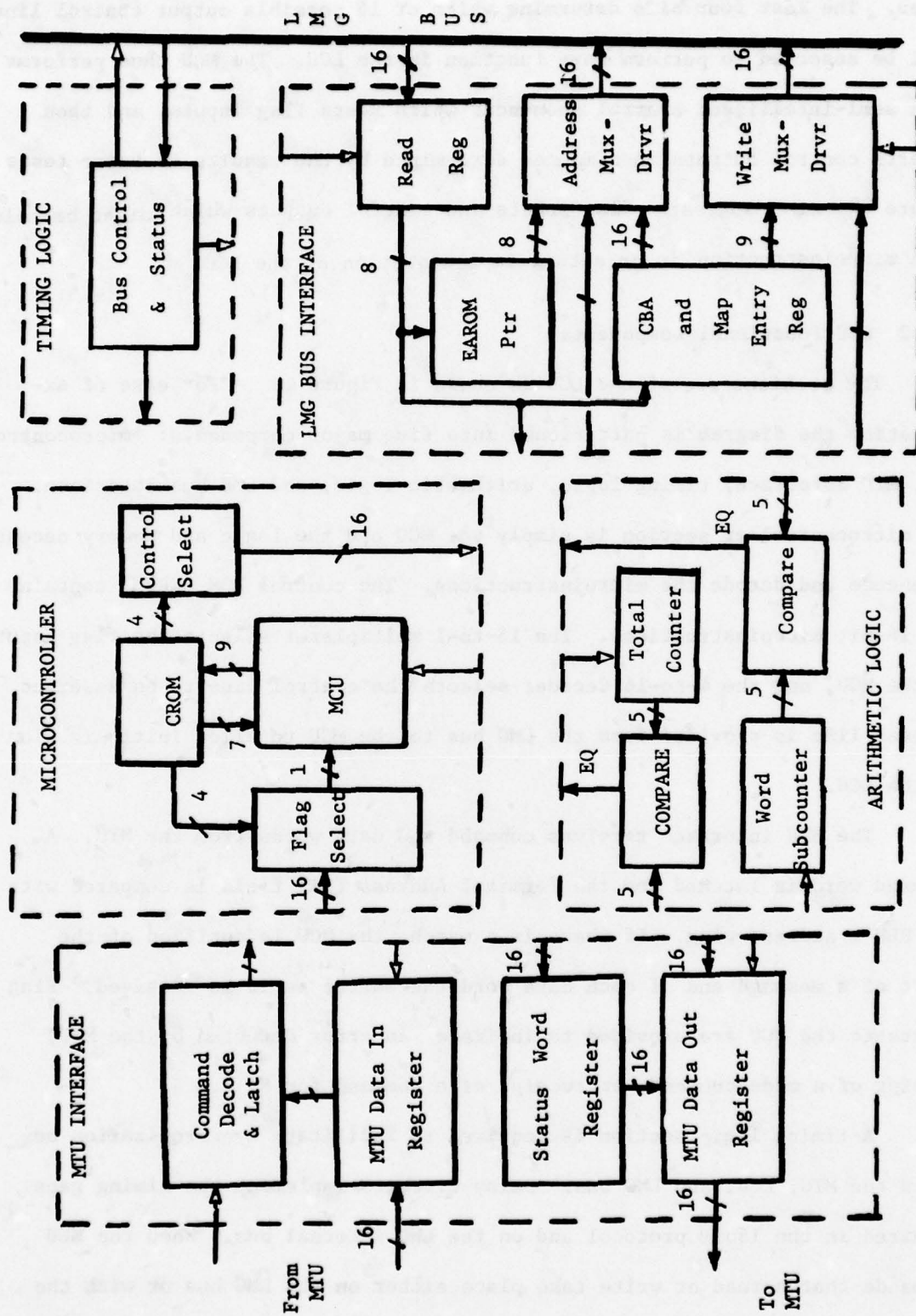


Fig. 40 Basic Architecture of the LCU

MTU, the timing logic translates this request into a properly timed control signal. Likewise, control signals received from the LMG bus or MTU are latched by the timing logic long enough for the MCU to detect.

The arithmetic logic section performs the few arithmetic functions required. As each part of a message is sent to or received from a particular LM/LA, a counter is incremented until the count equals the word subcount (WSC) in the map entry obtained from the mapping EAROM. A comparator provides the corresponding flag input to the MCU. A second counter counts all of the words sent or received, the sum of all the WSC's, and this total is compared with the word count field of the command word. Inequality of the two is detected by the MCU as an error condition and the entire message is invalidated.

The LMG bus interface section implements all connections to the LMG bus. A 16-bit register is provided to latch data read from the bus until the MCU can route it further. An 8-bit counting register receives the list pointer obtained from the mapping EAROM, and is incremented as the MCU steps through the map entries for each message. Each map entry is stored and decoded in a second 16-bit register. This register is also used to hold the Command Buffer Address (CBA) pointer for messages to/from SA 1 (see Sec. 6.2.6.). The initial value is obtained from a dedicated RAM location. Two 16-bit multiplexer-drivers are provided for applying data and addresses to the bus. Control outputs from the MCU select the type of address and the source of the data to be applied. Four general categories of addresses are placed on the LMG bus by the LCU's address multiplexer-driver. The categories are illustrated in Figure 41, and their use in LCU operations is described in the sections that follow.

6.2.3 Accessing the Mapping EAROM

In order to address the mapping EAROM, the high order 6 bits of the

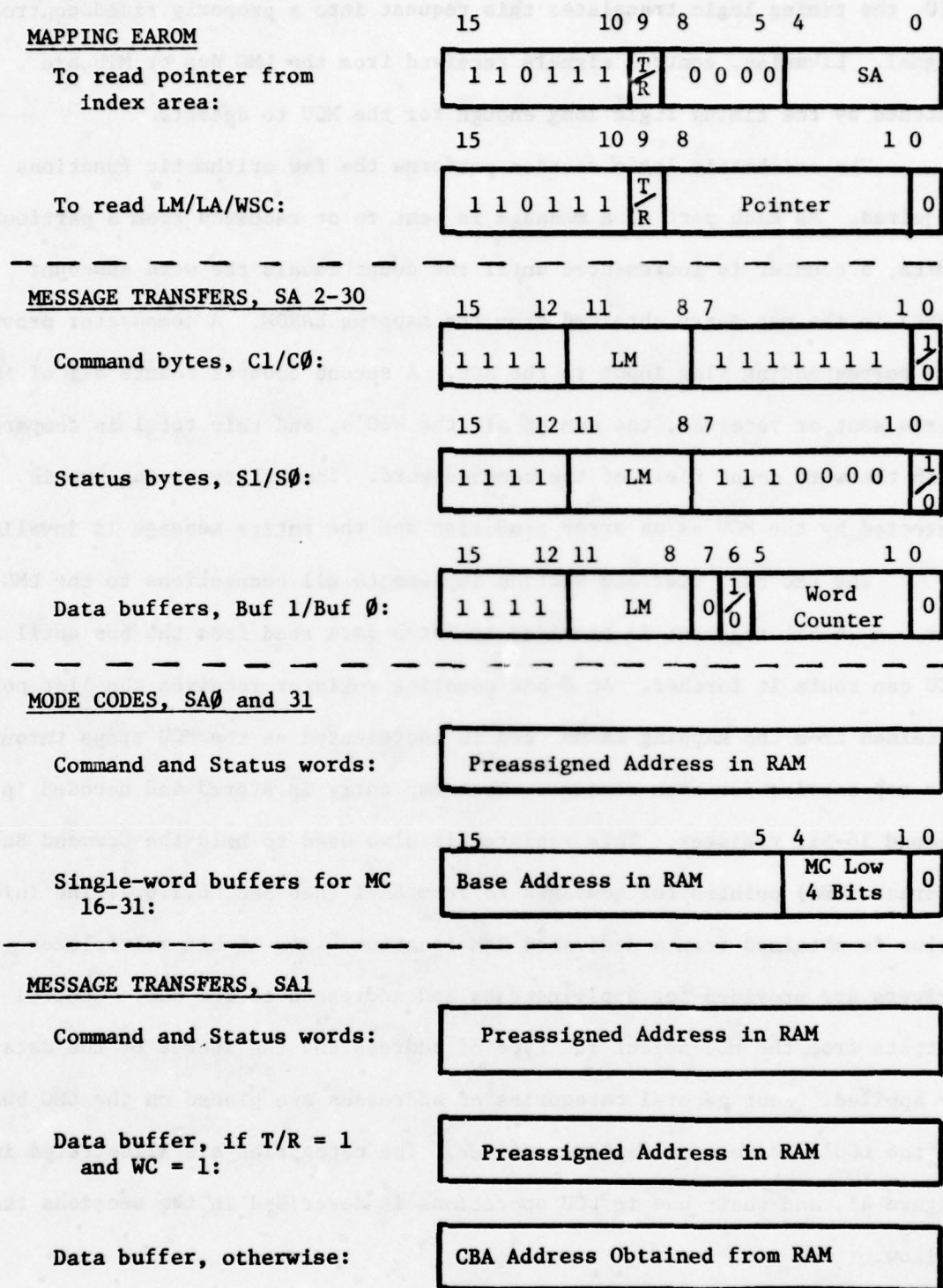


Figure 41 Addresses Used by LCU on LMG Bus

LMG bus address are set to 110111_2 . The low-order 10 bits choose which word or byte is to be read. The T/R bit from the DAIS command word is routed to bit 9 to select the upper or lower half of the EAROM, as was illustrated in Figure 37 .

When a transmit or receive command is received from the MTU, the SA and T/R bit are used to address one of the two index sections of the EAROM. As shown in Figure 41 , bits 5-8 of the address are set to zeroes, the SA is placed in bits 0-4, and a single-byte read is executed. The 8-bit pointer thus obtained is then routed to bits 1-8 in order to read the individual map entries for the message. This pointer is incremented through the EAROM entries until one of the entries has the EOM bit set, or until the message word count is exhausted, whichever comes first. If the two events do not occur at the same time, an error is indicated.

6.2.4 Data Transfer Operations

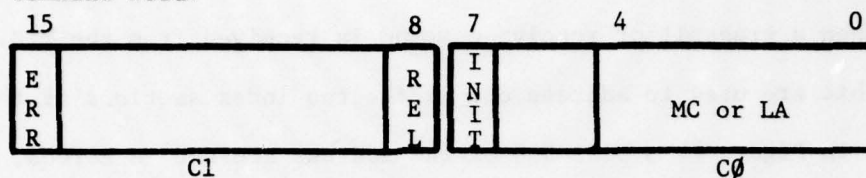
For command, status, and data transfers to and from shared memory, the LMG bus address is formed by setting bits 12-15 to all ones, and bits 8-11 to the LM number obtained from the mapping EAROM entry. The low eight bits are then selected by the LCU to address the command word, status word, or data buffers. Figure 41 illustrates the format for each.

The meaning of each bit in the command and status words, as described in Figure 13 , is repeated in a more general way for the LCU in Figure 42 .

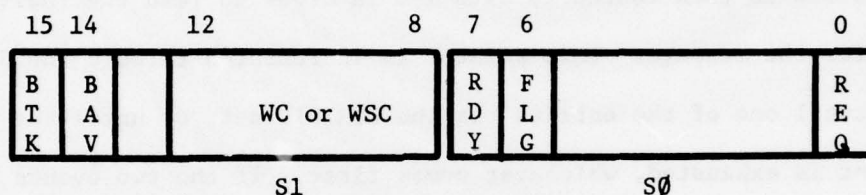
In executing a message transfer, the LCU first obtains a map entry from the mapping EAROM, as described in the previous section. If a valid (non-zero) map entry is obtained, and if the message is one to be transmitted by the RLU, the LCU next gives the status word to the MTU to transmit to DAIS. While the status word is being transmitted (transmit command) or while

The same command and status words, with slight variations, are used for mode commands (SA 0,31), LMG commands (SA 1), and data transfers to/from subsystems (SA 2-30).

Command Word:



Status Word:



Handshake Protocol:

- LCU sends C0, generating an interrupt
 - MC = 5-bit mode code (SA 0,31 only)
 - LA = 4-bit Link Address (SA 2-30 only)
 - INIT = 1 to initiate a data transfer
- LCU performs read-modify-write on S1, S0.
 - RDY = 1 if a buffer is available
 - BAV indicates which buffer (SA 2-30 only)
 - REQ = 1 to take the buffer
 - BTK indicates which buffer is taken (SA 2-30 only)
 - WC = word count (SA 1 only)
 - WSC = word subcount (SA 2-30 only)
 - FLG = 1 if subsystem is flagged as down (SA 2-30 only)
- LCU transfers data.
- LCU sends C1, generating an interrupt
 - REL = 1 to release buffer and interrupt LM or LMG
 - ERR = 1 if an error was encountered

Figure 42 Message Transfer Handshake Protocol

the first data word is being received (receive command), the LCU begins the data transfer handshake with the appropriate LM. The command byte C0 is written to the LM, giving the LA being referenced for this portion of the message. The LCU then immediately initiates a read-modify-write on the entire status word, bytes S0 and S1. If the RDY bit is zero, the LCU terminates processing of this message and sets the busy bit in the 1553B status word to be transmitted to DAIS. If RDY = 1, the LCU sets REQ = 1, sets BTK to whatever BAV is, inserts the WSC, and then executes the write portion of the read-modify-write. The LCU has had full control of the shared memory during the time so that there can be no intervening access to the Data Transfer status word by the LM.

The LCU now transfers as many data words as are indicated by the WSC field of the EAROM map entry. The value of the BAV bit in the status word is routed to bit 6 in addressing shared memory, bit 7 is cleared, and bits 1-5 contain the LCU's word counter so that consecutive locations in the SM buffer are addressed.

When the WSC count is exhausted, or when the original message word count is reached, the LCU stops transferring data. If only the WSC is exhausted, the LCU increments the mapping EAROM pointer, reads the next map entry, and repeats the process above. If, however, the EOM bit was set in the map entry just used and the message word count has been reached, it is time for the LCU to close out the message.

If the message was a receive message, the LCU gives the MTU the status word to transmit. The LCU then writes command byte C1 to the shared memory interface (SMI). Since the low byte of the address for C1 is all ones, the SMI will write it to all LM's, not just the one indicated by bits 8-11. The

ERR bit of C1 will be cleared by the LCU if no errors were encountered during the message transfer operation. If any errors were encountered, the ERR bit will be set.

If the command from DAIS was a transmit command, and if the LCR bit is set in the mapping EAROM entry, the LCU will obtain the Last Command Register from its RAM location and transmit it as the first data word.

6.2.5 Mode Code Operations

A mode command is indicated when the SA of the received command word from DAIS is 0 or 31. For simplicity of design the LCU's handshake with the MGP is substantially the same as it was with the SMI. See Figure 42 once again. Two locations in MGP RAM are set aside for the mode code command and status words. Upon receipt of a mode command from DAIS, the LCU writes the 5-bit mode code (instead of a 4-bit LA) into the low byte of the command word, interrupting the MGP. The LCU then does a read-modify-write on the status word. If RDY = 1, the REQ bit is set when the LCU writes the status word back. If RDY = 0, then the MGP is unable to respond to the mode command, and the LCU will set the busy bit in the status word returned to DAIS. Status byte S1 is ignored during mode operations.

Mode codes 16-31 require a single data word to be transmitted or received. Therefore, 16 consecutive locations in RAM are dedicated for these 16 words. If the mode code is among 16-31, the LCU performs the single word transfer using a RAM address formed as shown in Figure 41.

Once the status word has been given to the MTU to transmit, the LCU completes the handshake by writing command byte C1 to RAM, indicating whether any errors have been encountered.

6.2.6 SA 1 Operations

Messages for SA 1 are referred to the MGP, rather than the LM's. Again, the data transfer handshake is the same as for other subaddresses. However, the addresses for the command word, status word, and data buffer are obtained differently, as illustrated in Figure 41 .

The command and status words are given fixed, dedicated locations in RAM; and the protocol involving them is the same as it is for other subaddresses, except that what is written to command byte C0 is unimportant. Only the resulting interrupt to the MGP is required.

The LCU transfers the message using one of two sources for the buffer address. If the message is a receive message with a word count of one, a preassigned address in RAM is always used. If the message is other than a single word to be received, the LCU obtains the buffer starting address from a special RAM location called the Command Buffer Address (CBA), moves it to a local register, and increments it until the word count is exhausted. The MGP is presumed to have loaded the CBA location previously with the correct buffer address.

6.2.7 Status Word

For any of the operations described above, the LCU must transmit the bus status word to DAIS in response to the received command word. Maintenance of bits in the status word is shared by the LCU and MGP, but the LCU has sole responsibility for its transmission.

In a preassigned location in RAM, the MGP continually updates two status bits: asynchronous service request bit and terminal flag bit. The service request bit is set by the SM Handler whenever a request is received

from an LM and is placed into the Asynchronous Request Queue (ARQ). The service request bit is cleared by the Mode Code Processor task when the last request in the ARQ has been removed and serviced. The terminal flag bit is set by the MGP whenever it wishes to indicate an RLU fault condition.

In its own status word register the LCU sets three other bits according to conditions existing during execution of a bus command: message error bit, busy bit, and subsystem flag bit. The message error bit is set when the MTU indicates irregularities in the command word or the data words. The busy bit is set if the RDY bit in the Data Transfer Status is not set for one of the LA's involved in the message. The subsystem flag bit is set if the FLG bit in the Data Transfer Status is set, indicating a subsystem failure condition.

Whenever the LCU must transmit the status word, it reads from the preassigned RAM location to get the MGP's two bits, merges them with its own bits, and then transmits the status word.

6.3 SOFTWARE ORGANIZATION

The operations within the LMG are directed by a real-time executive which has as its primary functions task scheduling, task completion analysis, and interrupt processing. Other system functions are performed by individual system tasks. Table 10 lists all system tasks and programs, with details on what functions they perform and how they are invoked (which task, or which LMG command through SA 1). Table 11 lists the system tables with information on their content, which tasks constructed them, and which tasks reference them. Details of the executive and system software are given in Appendix C. More information on the functions of the system tasks is given

TABLE 10
LMG SYSTEM TASKS

<u>NAME</u>	<u>FUNCTIONS</u>	<u>INVOKED BY</u>
Initialization	<ul style="list-style-type: none"> - initialize the LMG and all LM's - load the mapping EAROM - make specific changes in the RLU configuration - modify the mapping EAROM 	<ul style="list-style-type: none"> - INIT command - CONFIG command - EAR task
LMG Diagnostic	<ul style="list-style-type: none"> - perform diagnostics on MGP, memory, LCU, and MTU - request LM's to run diagnostics 	<ul style="list-style-type: none"> - MGDIAG command - EAR task
LMG Command Processor	<ul style="list-style-type: none"> - respond to messages to SA 1 as LMG commands - execute LMG commands, invoking other tasks as necessary 	<ul style="list-style-type: none"> - SA 1 interrupt from LCU
Maintenance Support	<ul style="list-style-type: none"> - execute commands given from maintenance port (MP), invoking other tasks as necessary - provide debug and system monitoring services 	<ul style="list-style-type: none"> - command from MP
Error Analysis and Recovery (EAR)	<ul style="list-style-type: none"> - respond to LM Status Alert changes (other than asynchronous service request) - invoke diagnostics on LMG or LM, as necessary - reconfigure backup subsystems, if required - maintain an System Error Log in EAROM 	<ul style="list-style-type: none"> - ERROR command - SM Handler on Status Alert change - EXEC (on non-zero task completion)
Mode Code Processor	<ul style="list-style-type: none"> - execute all mode code operations 	<ul style="list-style-type: none"> - mode code interrupt from LCU

TABLE 10
LMG SYSTEM TASKS (CONT'D.)

<u>NAME</u>	<u>FUNCTIONS</u>	<u>INVOKED BY</u>
Local Monitor	<ul style="list-style-type: none"> - regularly monitor communications with DAIS - initiate local processing tasks according to preset criteria, or upon command 	<ul style="list-style-type: none"> - LOCAL command
Program Load	<ul style="list-style-type: none"> - receive local processing tasks from DAIS - facilitate transfer of programs from DAIS to the LM's 	<ul style="list-style-type: none"> - PRGMLOD command
Table Transfer	<ul style="list-style-type: none"> - transfer LMG tables to DAIS - facilitate transfer of tables from LM's to DAIS 	<ul style="list-style-type: none"> - XFRTBL command
SM Handler	<ul style="list-style-type: none"> - transfer data to/from SM - respond to interrupts caused by Status Alert changes - put asynchronous service requests into the ARQ 	<ul style="list-style-type: none"> - most system tasks
LCU Handler	<ul style="list-style-type: none"> - respond to interrupts from LCU 	<ul style="list-style-type: none"> - LMG Command Processor task - Mode Code Processor task

TABLE 11
LMG SYSTEM TABLES

<u>NAME</u>	<u>CONTENTS</u>	<u>AUTHOR TASK</u>	<u>READER TASKS</u>
Link Module Directory (LMD)	By LM: directory of all subsystems interfaced to each LM.	Initialization, Configuration	Configuration
RLU Configuration Table (RCT)	By EID: List of all EID's required by DAIS, LM/LA of corresponding subsystem, asynchronous request vectors	Initialization, Configuration	Configuration SM Handler
Local Program Directory (LPD)	Directory of all Local Processing tasks received from DAIS	Program Load	Local Monitor
System Task Directory (STD)	By Name: pointers to all system tasks	-	EXEC
Active Task List (ATL)	Linked lists of all active tasks, pointers to them, task states	EXEC	EXEC
System Error Log	In EAROM, a log of all errors discovered by the EAR task	Error Analysis and Recovery	Maintenance Support

in the sections below.

It should be pointed out that there are no data transfer tasks to handle messages communicated between DAIS and the subsystems. These messages are all handled by the LCU using DMA transfers to/from the LM's. LMG software is involved only in the sense that message errors may cause the LMG to respond to subsequent error processing inquiries from the master executive. In addition, the MGP maintains two status word bits, as described in Section 6.2.7.

6.3.1 SA 1 Processing - LMG Commands

Special functions available in the MGP are invoked by the master executive through messages to SA 1. When a one-word message to SA 1 is received, the LCU always places the word in a fixed location in MGP RAM and interrupts the MGP. The interrupt service routine (ISR) for the LMG command processor task immediately examines the word to see what command has been issued by the master executive. In some cases quick action may be taken by the ISR; in other cases system tasks will be scheduled. A list of the commands available is given in Table 12.

If the command to SA 1 indicates anything other than a single-word receive message, the LCU obtains from a special location in RAM the address of the source or destination data buffer. This location is called the Command Buffer Address (CBA) and has been previously loaded by the LMG Command Processor task.

As an example of how this procedure is used, suppose the master executive wishes to read the directory of one of the LM's from the MGP's LMD Table. A XFRTBL command would be sent to SA 1 as a single-word receive

TABLE 12
LMG COMMANDS

- RESET - halt all system activity, stop all tasks and remove from Active Task List, reset all status, reset all LM's.
- INIT - run Initialization task to execute RLU initialization sequence.
- CONFIG - run Configuration task to modify the present RLU configuration.
- MGDIAG - run LMG Diagnostic on processor, memory, LCU and MTU; run LM diagnostics.
- PRGMLD - run Program Load task to receive local processing programs from DAIS, or to transfer programs to LM.
- XFRTBL - run Transfer Table task to transfer tables from LMG or LM to DAIS.
- LOCAL - run Local Monitor task and pass it criteria for initiating local processing.
- ERROR - run Error Analysis and Recovery task to analyze errors indicated to DAIS in the status word or the BIT word.
- RUN - initiate operation of all or selected subsystems.
- STOP - stop operation of all or selected subsystems.

message; and it would be followed immediately by a transmit command, with an appropriate word count, to SA 1. During the time period between the two messages, the ISR interprets the XFRTBL command and loads the CBA with a pointer to the requested LM's directory in the LMD table. The desired information is thus quickly and efficiently requested and given.

As a variation, suppose the request is one which the ISR cannot immediately satisfy. As an example, the master executive may wish to read a table in one of the LM's. In this case the ISR would simply schedule the Transfer Table task to request the information from the LM. When the table is ready for transmission, either in an LMG RAM buffer, or perhaps still in the shared memory LM buffer itself, the Table Transfer task would issue an asynchronous service request for SA 1 and load the CBA with the address of the buffer. The master executive would respond to the service request by issuing a transmit command to SA 1 to obtain the table.

Other variations on these procedures allow a powerful and flexible capability for performing special functions for the master executive, or for the transfer of data between the master executive and the LMG.

6.3.2 Initialization and Configuration

Upon receipt of the INIT command, the LMG will enter the initialization sequence for the LM's. A flow diagram for this procedure is given in Figure 43 .

During this procedure, the initialization task obtains the information necessary for constructing the map entries for each message, and then loads the mapping EAROM.

Since the INIT command causes the entire initialization sequence to begin, a CONFIG command is available for making alterations to the existing

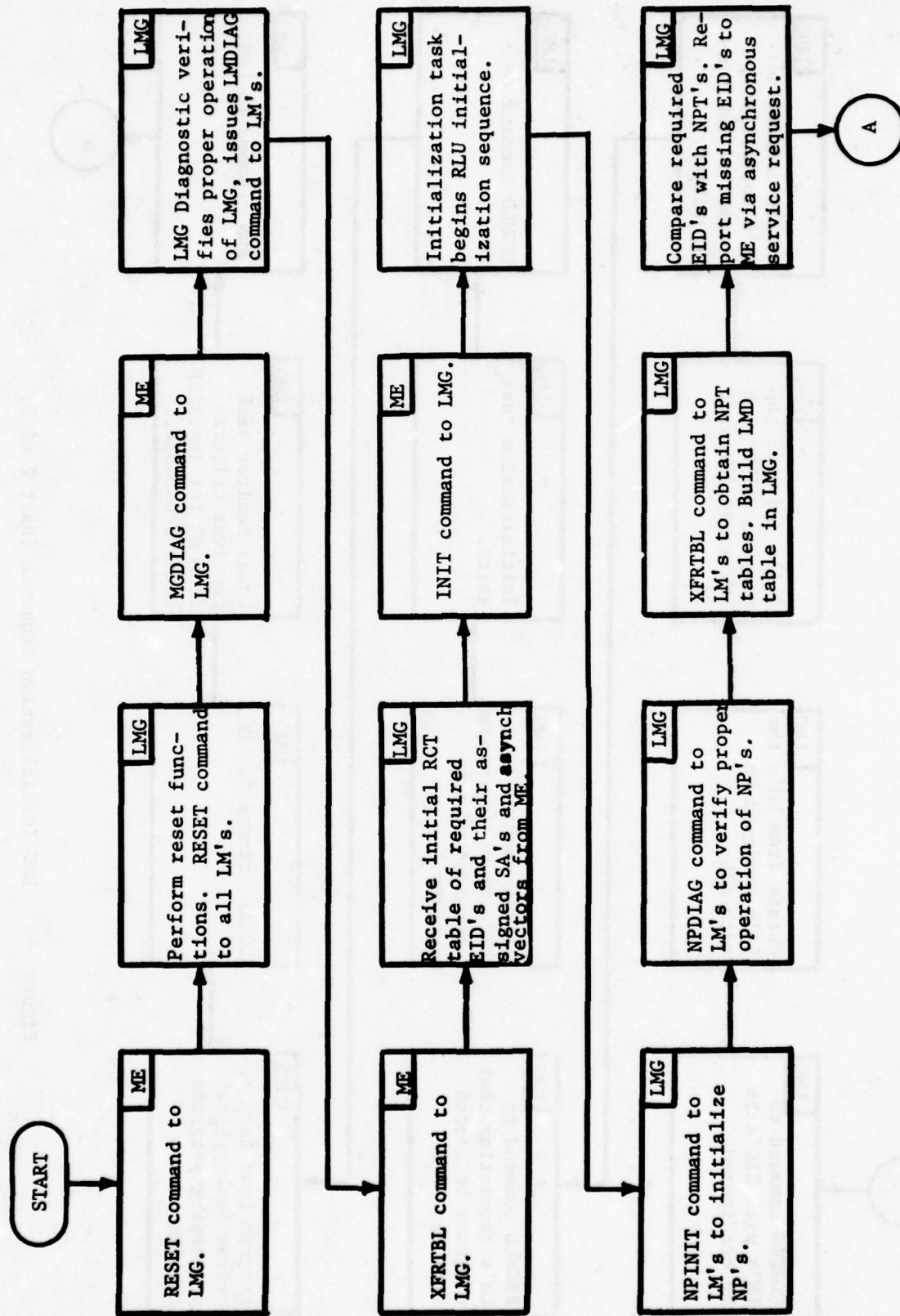


Figure 43 LMG Initialization Sequence (Part 1 of 3)

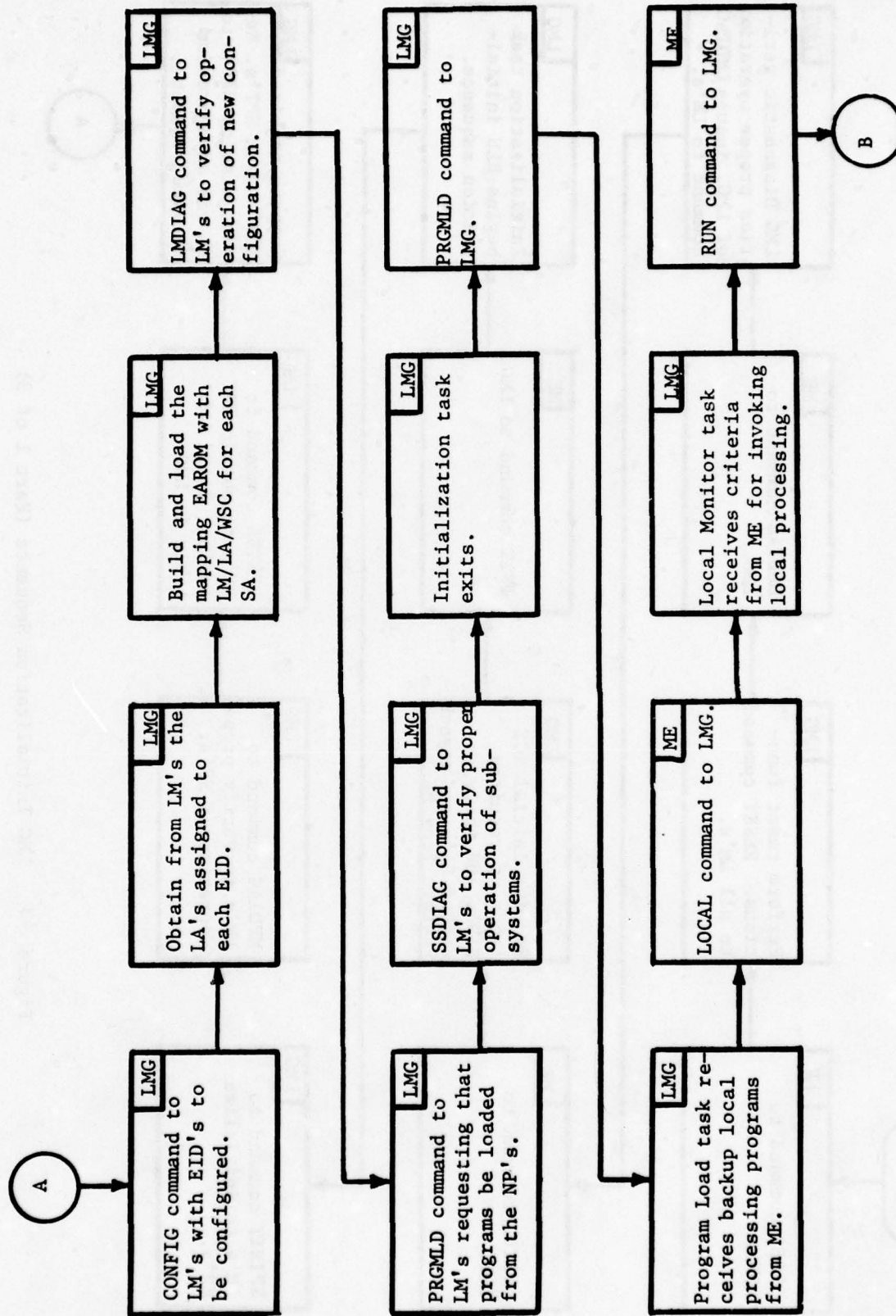


Figure 43 LMG Initialization Sequence (Part 2 of 3)

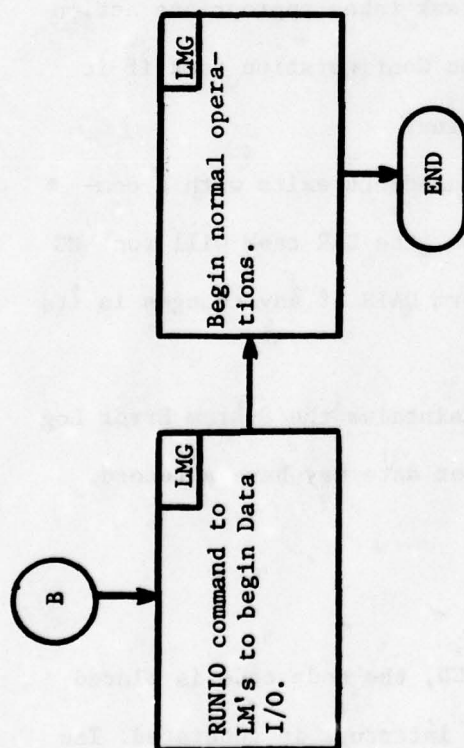


Figure 43 LMG Initialization Sequence (Part 3 of 3)

LMG configuration. For example, the master executive may choose to take a particular subsystem off-line, or swap it out with a backup subsystem in another LM. The CONFIG command would invoke the configuration task to make these changes and to alter the mapping EAROM accordingly. The configuration task can also be invoked internally by the Error Analysis and Recovery task to make selective changes in the system configuration, transparent to the master executive.

6.3.3 Error Analysis and Failure Management

All error analysis and failure management is directed by the Error Analysis and Recovery (EAR) task. When the SM Handler responds to an LM Status Alert interrupt, if the new status indicates a possible error condition in the LM, the EAR task is invoked. The task takes appropriate action such as running LM diagnostics or initiating the Configuration task if it decides a backup subsystem must be brought on-line.

If one of the MGP's own system tasks or handlers exits with a completion status that indicates an error condition, the EAR task will run LMG diagnostics, reconfigure as necessary, and inform DAIS of any changes in its own status.

No matter what the error, the EAR task maintains the System Error Log in EAROM so that maintenance personnel at a later date may have a record of all system failures and their known causes.

6.3.4 Mode Code Processing

When a mode command is received by the LCU, the mode code is placed in a fixed location in MGP RAM, and a mode code interrupt is generated. The Mode Code Processor task is activated and examines the mode code to determine

the action required. If none is required, for example, if the LCU is simply to transmit the status word, then the task terminates. Otherwise the task will perform the required operation, such as clearing a service request (MC = 17).

Mode codes 16-31 require the receipt or transmission of single data words. Sixteen consecutive locations in RAM are dedicated for this purpose, and the LCU automatically accesses these locations. The Mode Code Processor is responsible for disposing of words received from DAIS. Several system components are responsible for seeing that the words to be transmitted are kept updated. For example, the Mode Code Processor insures that the Activity Register (MC = 16) always has the highest priority item in the Activity Request Queue (ARQ); the Mode Code Processor and the EAR task together keep the BIT word (MC = 19) updated; and the LCU itself updates the Last Command Register (MC = 18).

6.3.5 Asynchronous Service Requests

Figure 44 diagrams the manner in which asynchronous service requests are handled in the RLU. When a Status Alert interrupt to the MGP indicates an asynchronous service request, the LMG's SM Handler obtains the request vector corresponding to the requesting LM/LA from the RLU Configuration Table. It then calls the system queueing routine which oversees the Asynchronous Request Queue (ARQ). This routine will add the new vector to the ARQ, insure that the Activity Register RAM location has the next vector out of the ARQ, and set the service request status bit.

When mode command 16 is received, the LCU automatically transmits the Activity Register to DAIS. The master executive inspects the activity

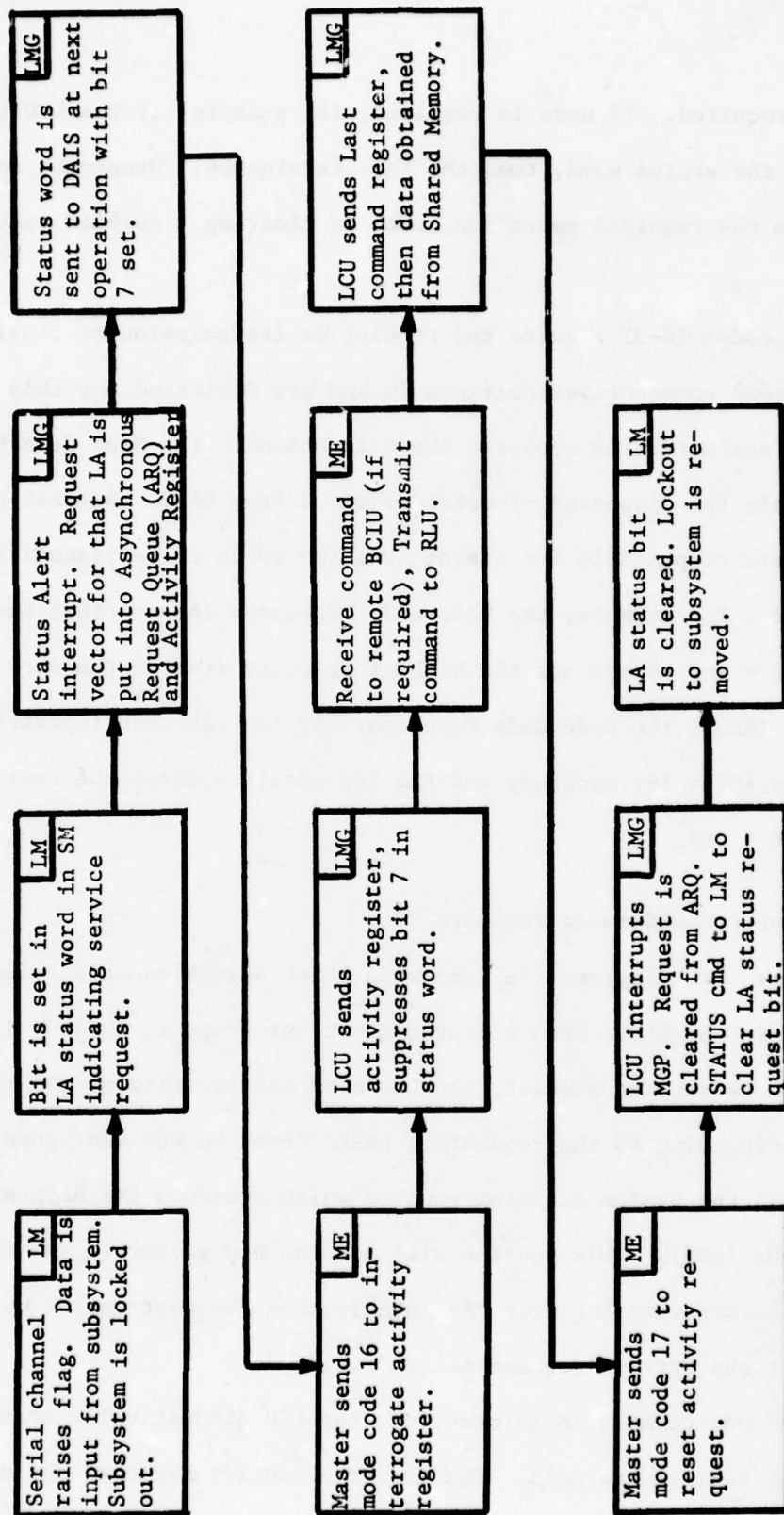


Figure 44 Asynchronous Message Operation (LMG and LM)

register value, and then sends a transmit command to the appropriate sub-address. When the message has been successfully transferred, the master executive sends mode command 17 to the RLU. The LCU interrupts the MGP, and the Mode Code Processor task is invoked. The task sends a STATUS command to the appropriate LM, asking it to clear its service request in the LA status word and to lift the lockout to the subsystem.

6.3.6 Local Processing

The LMG provides the capability of performing local control of subsystems if communications with the DAIS processor should lapse. The PRGLMD command to SA 1 is used first to download the programs which the LMG should execute. The programs necessarily come from the DAIS executive since they depend on the mission requirements. The MGP executive maintains the Local Program Directory (LPD) of all such programs.

The LOCAL command is then used either to provide the criteria under which the programs will be automatically invoked (for example, period of time without communication from DAIS), or to initiate immediate execution of the programs. When automatic invocation is requested, the Local Monitor task is scheduled to periodically monitor the communications activity. When a period of no activity has passed which is longer than the established criteria, the Local Monitor task will activate the appropriate local processing tasks.

6.3.7 Maintenance Support

The Maintenance Support task provides a wide range of capabilities for controlling and monitoring the entire RLU operation. All commands available through SA 1 can be issued through the maintenance port, in addition to

standard debug functions such as reading from any memory location. Thus, local processing tasks can be loaded and initiated, and data transfers through shared memory can be performed without using the LCU or MTU.

Or the Maintenance Support task can simply monitor specific actions taking place while on-line to DAIS. For example, the data transfer status word could be continually monitored to determine when a particular LM/LA is addressed; or the mode code command word could be monitored for all mode code operations.

All capabilities available from the MaintenanceSupport task can be accessed through the maintenance port, either manually from a CRT or by an automatic test computer.

SECTION VII

SUBSYSTEM SOFTWARE

The subsystem designer is responsible for developing programs for data conversion (DCP), and for subsystem initialization, error analysis, and diagnosis (SDM). These programs operate under control of certain Link Module (LM) tasks which are organized to relieve the subsystem programmer of most of the details at timing, interrupt processing, device handling, etc. Thus the programmer can concentrate on the problems of data conversion, error analysis, etc.

The present section provides an overview of the software operation as seen by the subsystem programmer. More discussion of the system software appears in Appendix C.

7.1 DATA I/O TASK OPERATION

Within an LM, data transfer is handled by data I/O tasks, each of which should be thought of as a pipeline from a data source to a data sink. Figure 45 is a partial schematic diagram of such a task. The operation is under control of two tables (ICT and OCT) and one subroutine (DCP) written by the user.

The two tables, the Input Control Table (ICT) and the Output Control Table (OCT) determine how the data source and sink handlers are to pass the data between the devices and the Task Input and Output Buffers, TIB and TOB, respectively; whether and how interrupts are to be serviced; and any other options relating to device operation. Each table also contains the identification of the handler to be used.

The user-written Data Conversion Program (DCP) passes information from

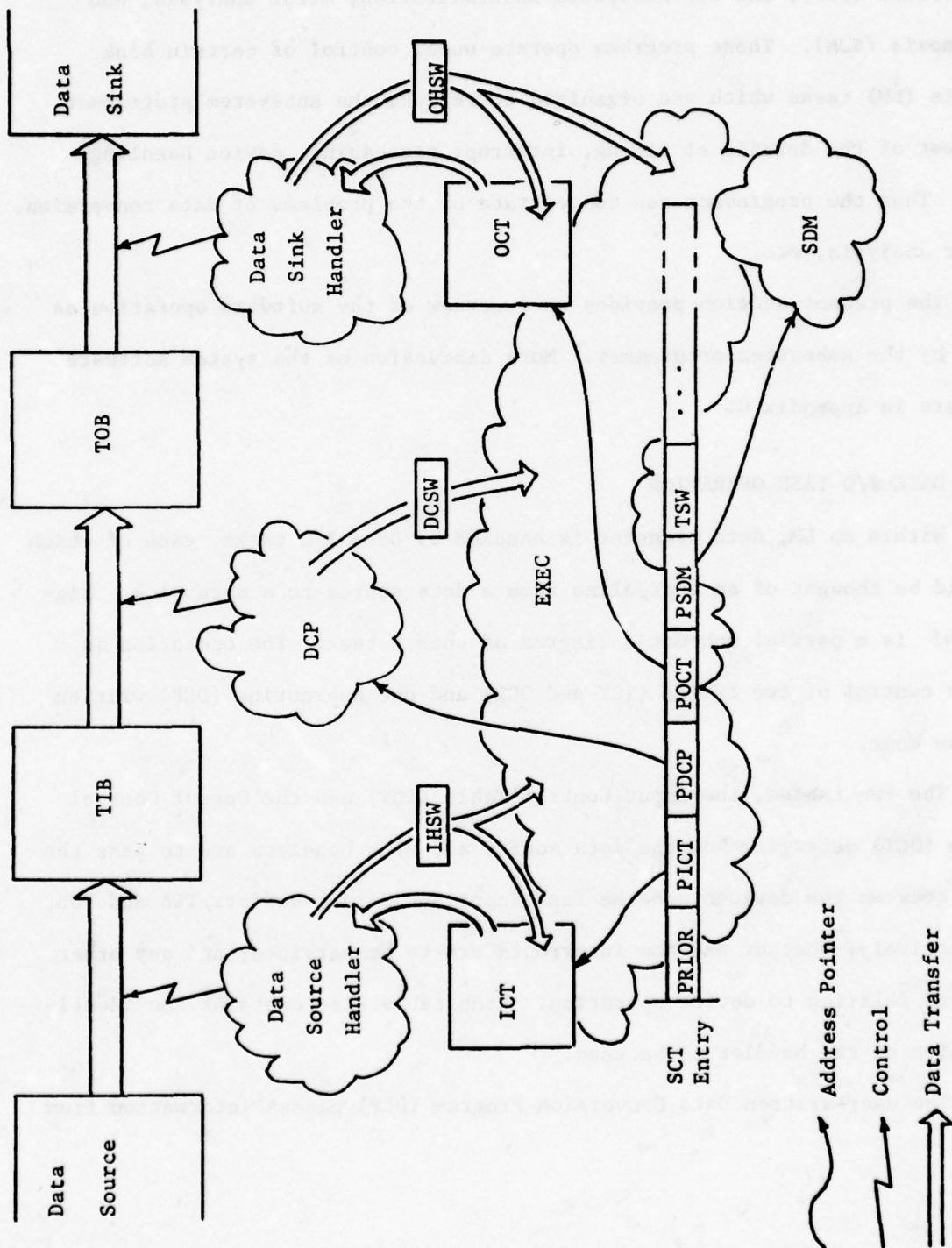


Figure 45 Data I/O Task

the TIB to the TOB. This program can perform data checks and conversions if these are needed. It can also send data to and receive data from other data I/O tasks through a shared portion of memory defined by the user. This data might be device status information, for example.

By setting the ABORT OUTPUT bit in the Data Conversion Status Word (DCSW) before returning to the calling program, the DCP can request that TOB not be passed to the Data Sink. This would be done in the case of a severe error, for example.

By setting the ERROR bit in DCSW before returning to the calling program, the DCP can request scheduling of the Subsystem Diagnostic Task (SDT). Normally this is done when errors are detected in the data: data source hardware errors (indicated to the DCP by the Input Hardware Status Word (IHSW), and data out-of-range conditions, for example. The SDT operates under control of a user-written program, the Subsystem Diagnosis Module (SDM). The SDT performs error analysis, error recovery, and error reporting functions defined by the user. This may include recording of the error condition in the subsystem nameplate. The requesting Data I/O Task is placed in the SUSPENDED state until the SDT terminates, at which time it is made ready or inactive depending on the condition of the REACTIVATE DCP bit of the Error Recovery Status Word (ERSW) set by SDP.

A Data I/O Task is identified by an entry in the Subsystem Configuration Table (SCT). This entry contains pointers to the ICT, OCT, DCP, and SDM for the task; the priority (PRIOR) of the task; a Task Status Word (TSW); and parameters used by the EXEC for scheduling the task.

7.2 SUBSYSTEM DIAGNOSTIC TASK OPERATION

The SDT is scheduled upon completion of the Configuration Task, upon termination of a Data I/O Task in which the ERROR bit of DCSW is set to 1, or upon receipt of a SSDIAG command by the SM. The method of scheduling is indicated by the parameter SDC (Subsystem Diagnostic Control), which is set to 0, 1, or 2, respectively. When SDT becomes active, the user written Subsystem Diagnostic Module (SDM) is called and SDC is passed to it.

As shown in Figure 46, SDM will normally comprise three parts: INIT, the subsystem initialization program; EAP, the Error Analysis Program; and SDP, the Subsystem Diagnostic Program. These three parts may, of course interact to whatever extent is necessary, but in any case the user has complete control of initialization, error analysis and recovery, and subsystem diagnosis.

7.3 DATA I/O TASKS: NAMEPLATE ENTRY ORGANIZATION

Within the nameplate for each subsystem there is an entry which contains the subsystem and task identifications; the control and configuration tables; and the DCP's and SDM's for the Data I/O Tasks. We will call this entry the Data I/O Task Entry. Parameters for the subsystem are in another nameplate entry, the Parameter Entry; this can be in Electrically Alterable Read Only Memory (EAROM) rather than in Read Only Memory (ROM) so the parameters can be changed (e.g., during subsystem calibration). The organization is shown in Figure 47 .

Each Data I/O Task is assigned a name and it has an entry in the SCT. The SCT entry contains pointers PDCP and PSDM, which are the addresses of the first words of the DCP for this task and the SDM, respectively, these

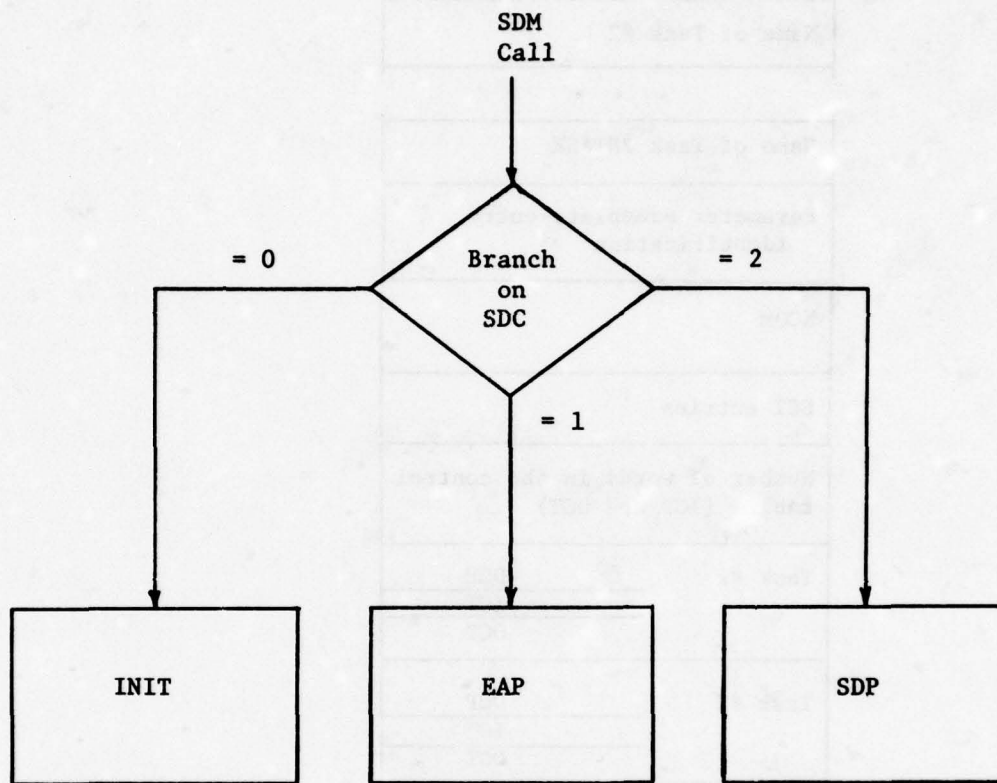


Figure 46 Typical Organization of SDM

Data I/O Task Entry

ID	
NTASK	
Name of Task #1	
Name of Task #2	
. . .	
Name of Task #NTASK	
Parameter Nameplate-entry Identification	
NCOM	
SCT entries	
Number of words in the control tables (ICT and OCT)	
Task #1	DCP
	ICT
	OCT
Task #2	DCP
	ICT
	OCT
. . .	
Task #NTASK	DCP
	ICT
	OUT
SDM	

Parameter Entry

Subsystem
Parameters

Figure 47 Nameplate Entries for Subsystem Data I/O Tasks

addresses being relative to the beginning of the nameplate entry. All programs are stored in interpretive form.

A word NCOM in the nameplate entry indicates the number of words of LM Memory that are to be allocated for common use by the subsystem of programs.

7.4 CONFIGURATION TASK OPERATION

When the Configuration Task is invoked (by a CONFIG command to the SM), it interrogates the Subsystem Information Channel (SIC) for subsystem Data I/O Task Entries. From each such entry it gets the subsystem task names, assigns a Link Address (LA), and sends the LA to the LMG. Then it relocates the task table entries, programs, and parameters to LM memory, and it assigns a common memory area. Finally, it schedules SDM with SDC = 0.

A flow chart for the Configuration Task appears in Figure 48, the arrangement of the user area of LM memory in Figure 49.

7.5 THE DEVICE CONTROL TABLES ICT AND OCT

The data source and data sink handlers are driven by the tables ICT and OCT, respectively. Each of these tables comprises two blocks: UNIV, which has the same form for all devices; and SPEC, which is dependent on the specific device. UNIV contains the following information:

AHAND	address of the device handles (user codes 0 for Null Handler, 1 for ICA Handler, and 2 for SM Handler),
IO	= 0 if input device (data source), = 1 if output device (data sink),
ABUF	address of the buffer,

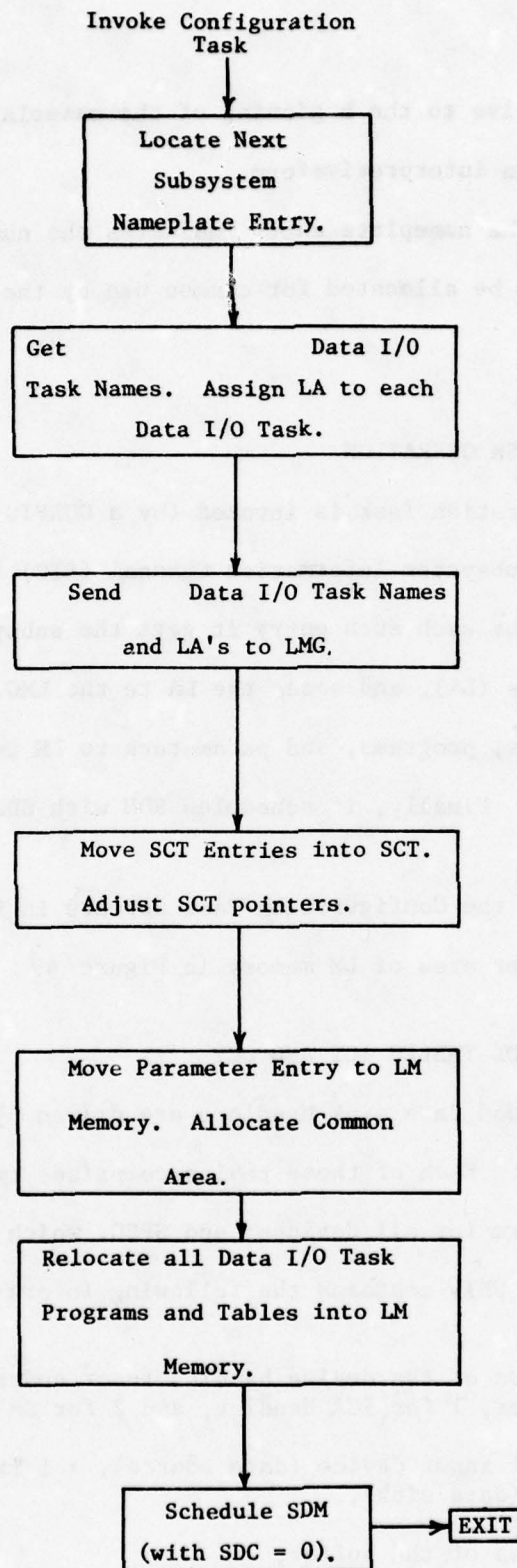


Figure 48 Configuration Task Operation

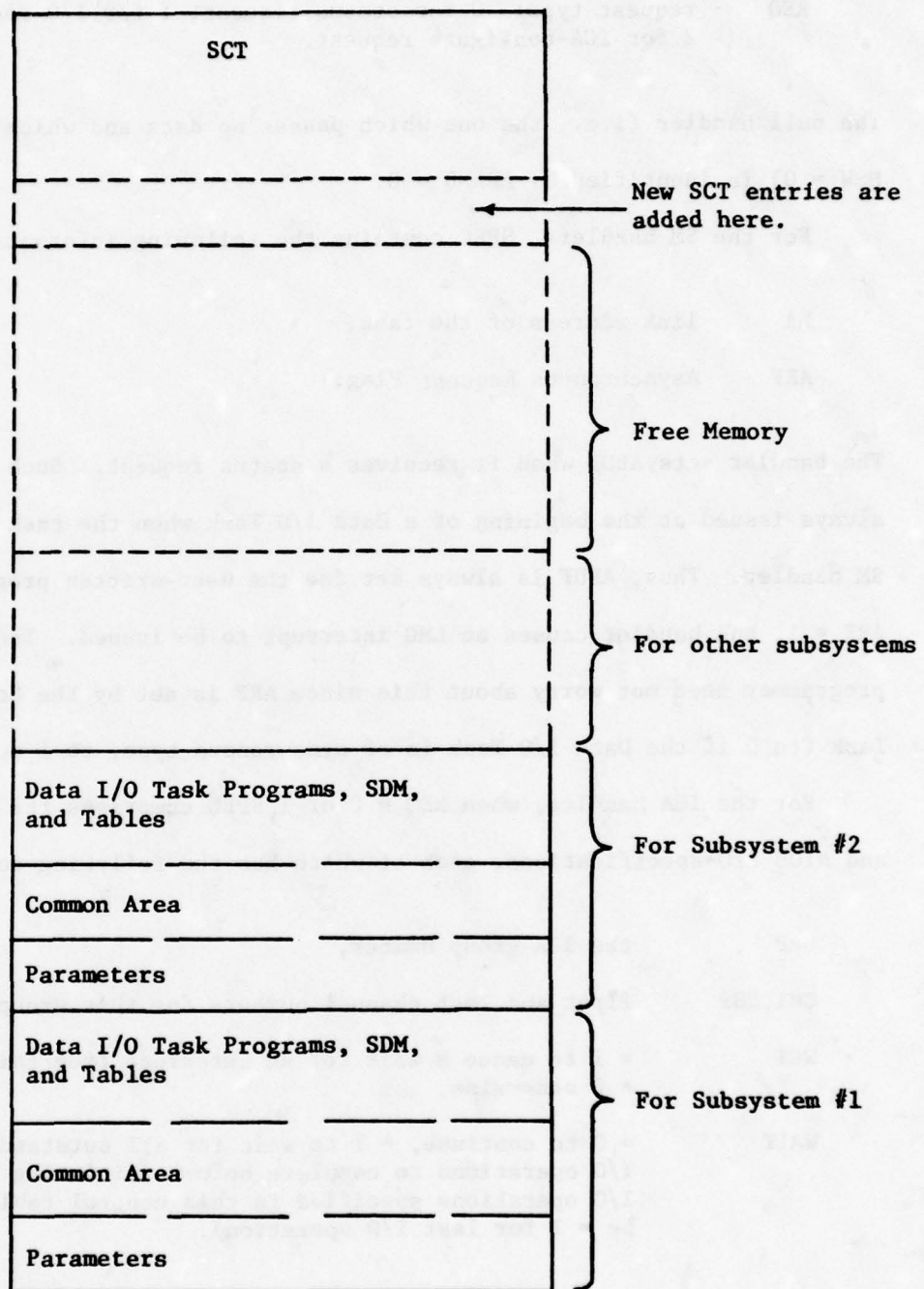


Figure 49 Allocation of LM Memory User Area

HSW hardware Status Word (set by the handler),
 REQ request type: 0 for status request, 1 for I/O request,
 2 for ICA-configure request.

The null handler (i.e., the one which passes no data and which always sets HSW = 0) is identified by AHAND = 0.

For the SM handlers, SPEC contains the following information:

LA link address of the task,
 ARF Asynchronous Request Flag.

The handler sets ABUF when it receives a status request. Such a request is always issued at the beginning of a Data I/O Task when the task involves the SM handler. Thus, ABUF is always set for the user-written programs. When ARF = 1, the handler causes an LMG interrupt to be issued. The subsystem programmer need not worry about this since ARF is set by the Configuration Task (to 0 if the Data I/O Task is of synchronous type, to 1 otherwise).

For the ICA handler, when REQ = 0 or 1, SPEC comprises the number NIOS, and NIOS I/O-specifications, each of which has the following form:

GRP the ICA group number,
 CH1,CH2 first and last channel numbers for this group,
 WGI = 1 to cause a wait for an interrupt from this group,
 = 0 otherwise,
 WAIT = 0 to continue, = 1 to wait for all outstanding ICA
 I/O operations to complete before initiating any more
 I/O operations specified in this control table (must
 be = 1 for last I/O operation).

The ICA handle initiates the I/O operations for the I/O specifications in

order until WAIT = 1 is encountered. It then waits until the I/O operations specified to this point have been completed. Then it continues to the next I/O specification, continuing in this way until all NIOS specifications have been processed. Words from the I/O operations are put into consecutive locations in the buffer in the order given by the I/O specifications.

When REQ = 2, the ICA handler configures the ICA. SPEC is empty. The buffer contains the 32 words of configuration parameters for the ICA followed by the five operational control words. The five operational status words are returned in the next five locations. The total length of the buffer is 42 words.

7.6 EXEC FACILITIES FOR THE USER

The system executive, EXEC, provides the user with the following facilities through an EXEC call with the parameter FACIL set as shown.

FACIL	Facility Description
0	Terminate the current task.
1	Suspend the current task and schedule task NTASK at priority PRIOR. NTASK may <u>not</u> be the current task.
2	Terminate the current task, then schedule task NTASK at priority PRIOR at time NTIME. NTASK may be the current task. [This is used by refresh-type Data I/O Tasks, for example].
3	Abort task NTASK immediately.
4	The current time is TIME (returned to the calling program).
5	Suspend the current task until time NTIME.

7.7 SUMMARY OF REQUIRED USER-WRITTEN SOFTWARE

For each subsystem the user must supply a nameplate containing the fol-

lowing software:

1. subsystem ID,
2. number of Data I/O Tasks,
3. names of the Data I/O Tasks,
4. identification of the nameplate entry containing the subsystem parameters,
5. the number of words of common memory needed,
6. the SCT entries for the Data I/O Tasks,
7. the tables ICT and OCT for the Data I/O Tasks,
8. the DCP for each Data I/O Task,
9. the SDM,
10. subsystem parameters.

Items 1-8 must be in a nameplate entry having the format described in Section 7.3. Item 10 must be in a separate nameplate entry. The nameplate must have entries for any special user-specified features, such as performance recording. User programs must schedule the Nameplate Handler Task for access to any nameplate entries.

7.8 A SUBSYSTEM EXAMPLE

In this section we present an example to illustrate how the RLU can be used to manage data from a subsystem. This example, in which the subsystem senses the angular position of a shaft, has been contrived to illustrate what can be done with the RLU, how it can be done, and what problems a subsystem developer must face. Though it one can see how the interface can be organized for function rather than for signals of the same type; how the LM can handle

such problems as data checking and conversion, subsystem calibration, error processing; and how the electronic nameplate is involved in the storage of subsystem programs, operational parameters, and performance records.

This example does not necessarily exemplify the best design for a position-measuring subsystem.

7.8.1 The Position Sensing Subsystem

The subsystem, shown schematically in Figure 50, comprises a resolver R, and four cam-operated switches S1, S2, S3, and S4. The resolver determines an angle θ' which it transmits in the form of three AC signals v_1, v_2, v_3 of amplitudes V_1, V_2 , and V_3 respectively. The true shaft angle θ is related to the angle θ' by

$$\theta = \theta' + \theta_0 + e,$$

where θ_0 is a constant and e is the error introduced by the resolver. We shall neglect e (i.e., assume that it is zero) from this point onward. The offset θ_0 can be determined during maintenance or by the calibration procedure described below.

Switches S1 and S4 are absolute limit switches which will be open except that S1 is closed when $\theta < \theta_{\min}$ and S4 is closed when $\theta > \theta_{\max}$ (see Figure 51). Switches S2 and S3 are calibration switches which are open except that S2 is closed for $\theta_{2\min} < \theta < \theta_{2\max}$, and S3 is closed for $\theta_{3\min} < \theta < \theta_{3\max}$. The nominal values of the angles are

$$\begin{array}{ll} \theta_{\min} = -45^\circ, & \theta_{\max} = 45^\circ, \\ \theta_{2\min} = -12^\circ, & \theta_{2\max} = -8^\circ, \\ \theta_{3\min} = 8^\circ, & \theta_{3\max} = 12^\circ. \end{array}$$

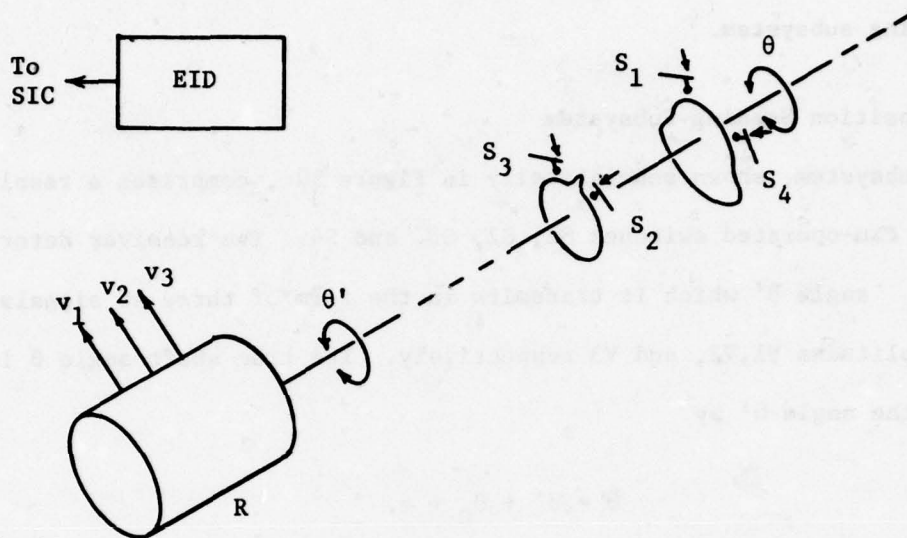


Figure 7.6 Position Sensing Subsystem

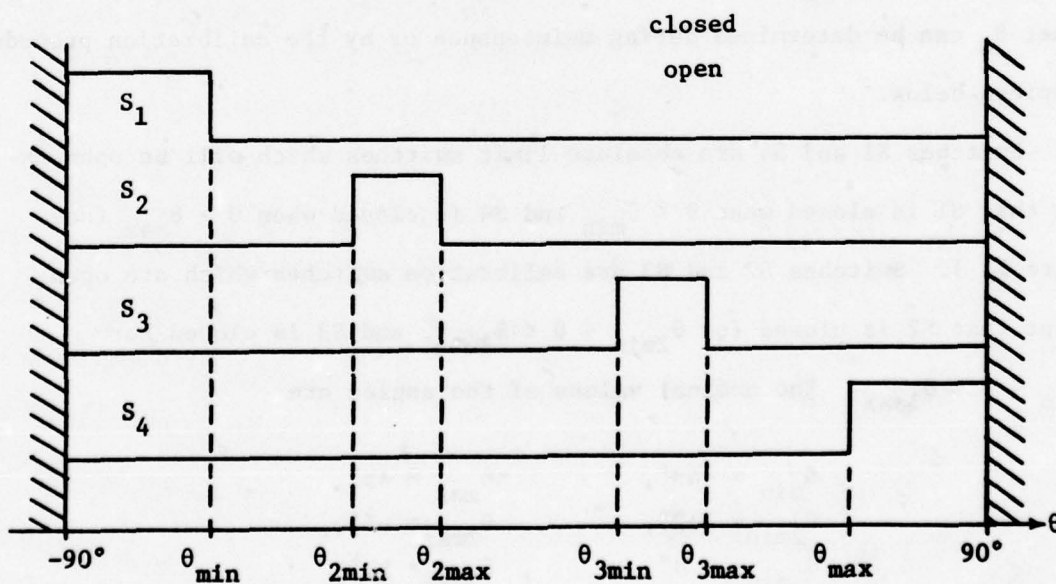


Figure 51 Switch Operation

Mechanical stops limit θ' to the range $-90^\circ < \theta < 90^\circ$.

Signals from the subsystem are taken to a Link Module (LM) as shown in Figure 52. Only half of the ICA channels are used for this subsystem.

7.8.2 Signal Handling

Three different types of information are to be handled by the LM. The shaft angle is encoded into the resolver signal amplitudes V_1, V_2 , and V_3 ; angular limit information is provided by the limit switches; and calibration points are determined by the calibration switches. As we shall see, the first two types of data are passed on to LMG/DAIS, but the calibration data is processed entirely within the LM.

7.8.2.1 Resolver Signals

Group A of the ICA is configured for AC A/D conversion, and the digitized values V_1, V_2 and V_3 are processed as described in Appendix D. This yields $(\cos \theta', \sin \theta')$ and resolver status information (from the tests of A and $V_1 + V_2 + V_3$). The pair $(\cos \theta, \sin \theta)$ is then determined according to

$$\begin{aligned}\cos \theta &= \cos(\theta' + \theta_0) = \cos \theta' \cos \theta_0 - \sin \theta' \sin \theta_0, \\ \sin \theta &= \sin(\theta' + \theta_0) = \sin \theta' \cos \theta_0 + \cos \theta' \sin \theta_0,\end{aligned}$$

and this pair and the status information are placed in SM for use by the LMG.

This operation is of the refresh type.

7.8.2.2 Calibration Data

Whenever one of S2 and S3 is closed, the calibration of the re-

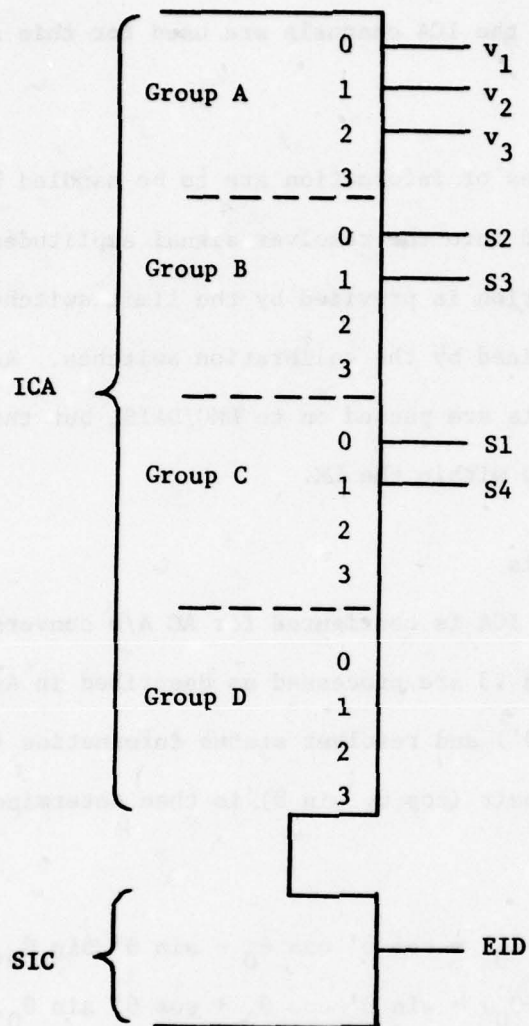


Figure 52 Subsystem Signals

solver is to be checked by determining whether the computed value of θ (as determined according to Section 2.1) is sufficiently close to the value of θ corresponding to the switch closure position. For the sake of accuracy, this check will be done only at the first turn-on point of one switch after the other switch has been on (i.e., at $\theta = \theta_{2\max}$ and at $\theta = \theta_{3\min}$, which we will call the calibration points of S2 and S3, respectively). If the deviation of the resolver-measured value of θ is significantly different from the value determined from the switch closure, the offset value θ_0 will be redetermined, and the new value passed to the electronic nameplate along with related performance information. The occurrence of this event will be signaled to LMG/DAIS via the status information mentioned in Section 7.8.2.1.

This operation is triggered by closure of S2 or S3.

7.8.2.3 Range Data

Closure of either of the range switches S1 or S4 indicates an out-of-range condition. This is reported immediately to the LMG by a status word placed in SM.

7.8.2 Error Processing

There are four ways in which there can be disagreement between the resolver and the calibration switches. To describe them we shall denote by r_2 and r_3 the values of θ determined by the resolver at the calibration points of S₂ and S₃, respectively. The four situations are:

$$D1. \quad r_2 = \theta_{2\max} \text{ and } r_3 \neq \theta_{3\min};$$

$$D2. \quad r_2 \neq \theta_{2\max} \text{ and } r_3 = \theta_{3\min};$$

D3. $r_2 \neq \theta_{2\max}$, $r_3 \neq \theta_{3\min}$, and $r_3 - r_2 = \theta_{3\min} - \theta_{2\max}$; and

D4. $r_2 \neq \theta_{2\max}$, $r_3 \neq \theta_{3\min}$, and $r_3 - r_2 \neq \theta_{3\min} - \theta_{2\max}$.

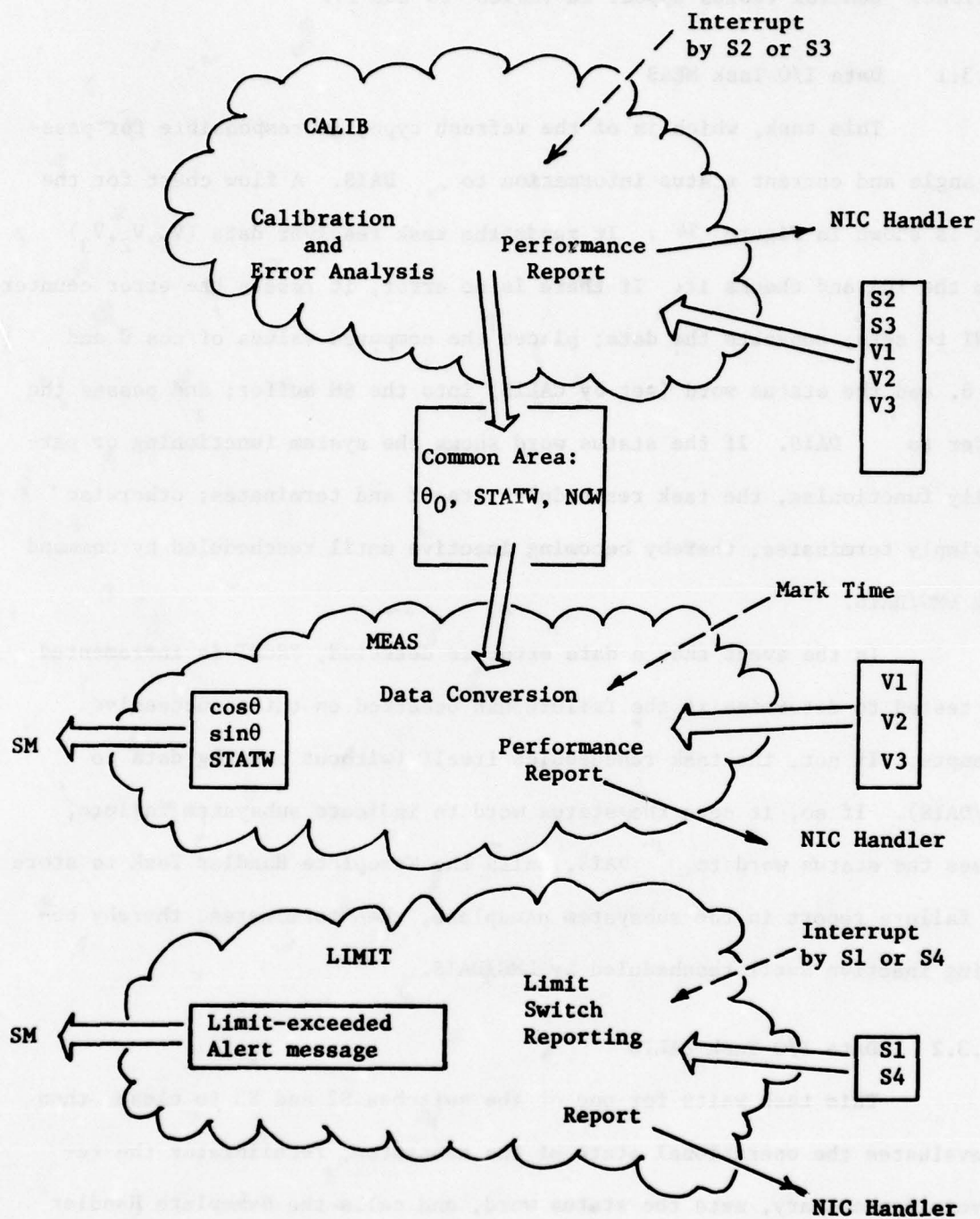
We will assume that D1 happens because of failure of S3, D2 happens because of failure of S2, D3 happens because of a change in θ_0 , and D4 happens because of failure of the resolver.* Thus, for D1 and D2, the subsystem software reports to DAIS the failure of the corresponding switch and ignores any future information from that switch. For D3, the software recalibrates the resolver by determining a new value of θ_0 . A report is made to DAIS only if the θ_0 differs from the maintenance-determined value by more than $\Delta\theta_{0\max}$, a quantity specified by the subsystem designer and stored as a parameter in the electronic nameplate of the subsystem. For D4, the software reports to LMG failure of the subsystem (the limit switches may still be operating of course).

Thus, there are three status levels: subsystem functioning, subsystem partially functioning (single error or excessive recalibration), and subsystem failed (D4). The latter two are to be reported to LMG/DAIS as soon as they are detected.

7.8.3 Software Tasks

The LM has three Data I/O Tasks: MEAS, CALIB, and LIMIT. The relations between them are indicated graphically in Figure 53. Initiation is done by the Configuration Task with user-written subroutine INIT in the

* Assumptions such as this can be made only after careful examination of subsystem failure modes and their influence on the whole system.



SDM. The function of each of these tasks is described in the following sections. Control tables appear in Tables 13 and 14.

7.8.3.1 Data I/O Task MEAS

This task, which is of the refresh type, is responsible for passing angle and current status information to DAIS. A flow chart for the task is shown in Figure 54. It reads the task resolver data (V_1, V_2, V_3) from the ICA and checks it. If there is no error, it resets the error counter ERCNT to zero; converts the data; places the computed values of $\cos \theta$ and $\sin \theta$, and the status word (set by CALIB) into the SM buffer; and passes the buffer to DAIS. If the status word shows the system functioning or partially functioning, the task reschedules itself and terminates; otherwise it simply terminates, thereby becoming inactive until rescheduled by command from LMG/DAIS.

In the event that a data error is detected, ERCNT is incremented and tested to determine if the failure has occurred on three successive attempts. If not, the task reschedules itself (without passing data to LMG/DAIS). If so, it sets the status word to indicate subsystem failure, passes the status word to DAIS, calls the Nameplate Handler Task to store the failure report in the subsystem nameplate, then terminates, thereby becoming inactive until rescheduled by LMG/DAIS.

7.8.3.2 Data I/O Task CALIB

This task waits for one of the switches S2 and S3 to close, then it evaluates the operational state of the subsystem, recalibrates the resolver if necessary, sets the status word, and calls the Nameplate Handler

TABLE 13
I/O CONTROL TABLES

<u>Data I/O Task</u>	<u>Table</u>	<u>Contents</u>	<u>Notes</u>
CALIB	ICT	AHAND = 1	For S2 and S3
		IO = 0	ICA
		ABUF = address of buffer	ICA buffer for CALIB
		REQ = 1	
		NIOS = 2	Two Groups
		GRP = B	WGI=1, WAIT=1 causes a wait for an interrupt from Group B be- fore reading the input of Group A Last I/O Operation
		CH1 = 0	
		CH2 = 1	
		WGI = 1	
		WAIT = 1	
		GRP = A	
		CH1 = 0	
		CH2 = 2	
MEAS	OCT	WGI = 0	
		WAIT = 1	
		AHAND = 0	Null
		IO = 1	
		ABUF = -	No buffer needed
		REQ = 1	
	ICT	AHAND = 1	For resolver
		IO = 0	ICA
		ABUF = address of buffer	ICA buffer for MEAS
		REQ = 1	
		NIOS = 1	
		GRP = A	Last I/O Operation
		CH1 = 0	
		CH2 = 2	
		WGI = 0	
		WAIT = 1	

TABLE 13
I/O CONTROL TABLES (CONT'D.)

<u>Data I/O Task</u>	<u>Table</u>	<u>Contents</u>	<u>Notes</u>
LIMIT	OCT	AHAND = 2	SM
		I/O = 1	
		ABUF = -	Set by SM Handler
		REQ = 1	
		LA = -	Set by configura- tion task
		ARF = -	Set by configura- task (=0)
	ICT	AHAND = 1	For S1 and S4
		IO = 0	ICA
		ABUF = address of buffer	ICA buffer for LIMIT
		REQ = 1	
		NIOS = 1	
		GRP = C	
		CH1 = 0	
		CH2 = 1	
		WGI = 1	
		WAIT = 1	Last I/O Operation
	OCT	AHAND = 2	SM
		IO = 1	
		ABUF = -	Set by SM Handler
		REQ = 1	
		LA = -	Set by configuration task
		ARF = -	Set by configuration task (=1)

TABLE 14
ICA CONTROL TABLE AND BUFFER
FOR THE CONFIGURATION TASK

Control Table	Notes
AHAND = 1	ICA
IO = 0	
ABUF = address of buffer	Configuration Buffer
REQ = 2	
Buffer	Notes
WORDS	
1-8 configure Group A for AC A/D	Resolver
9-16 configure Group B for switch contact closure input	Switches S2 and S3
17-24 configure Group C for switch contact closure input	Switches S1 and S4
25-32 any configuration	Not Used
33-37 command words	
38-42 status words	Returned

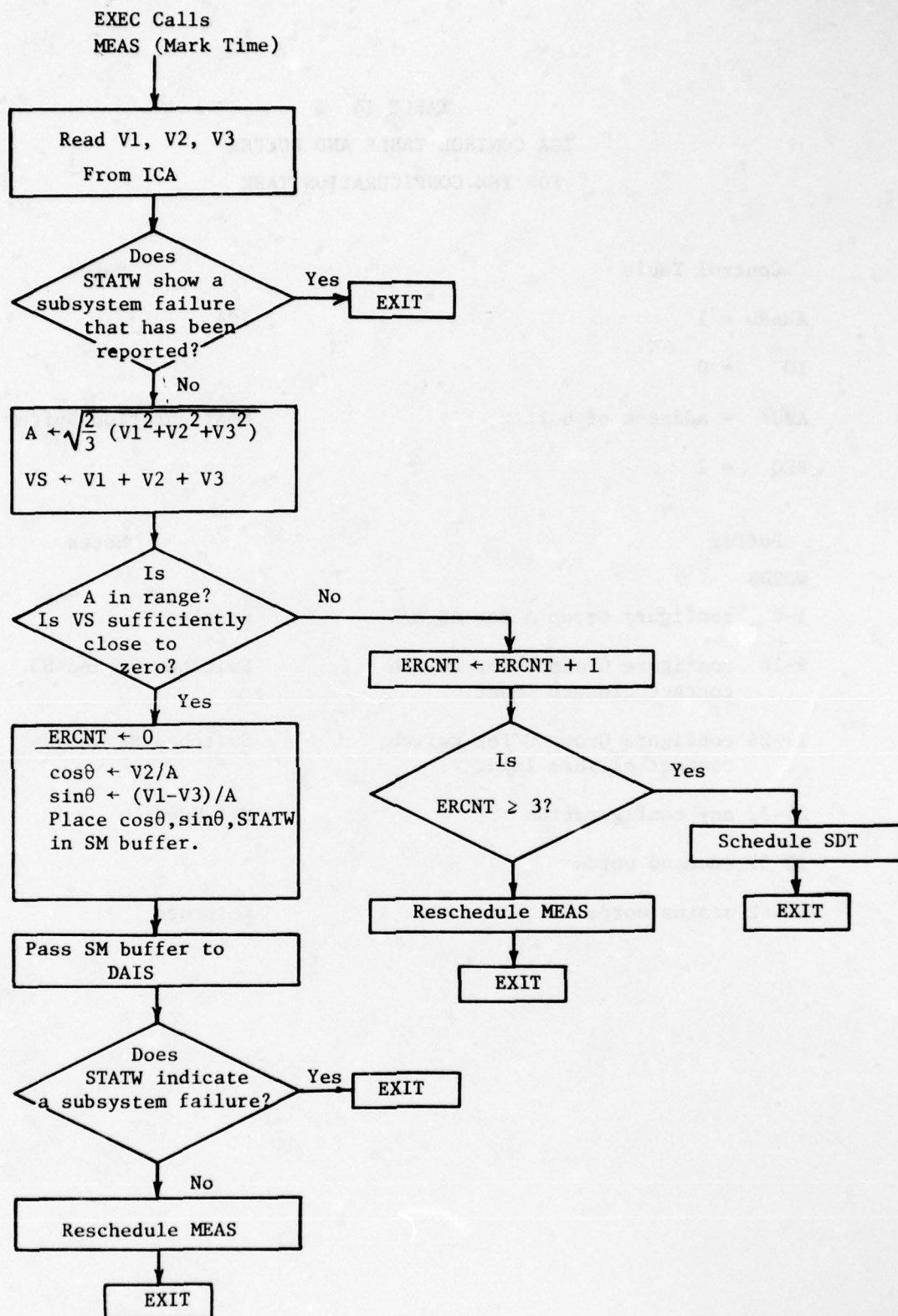


Figure 54 Task MEAS

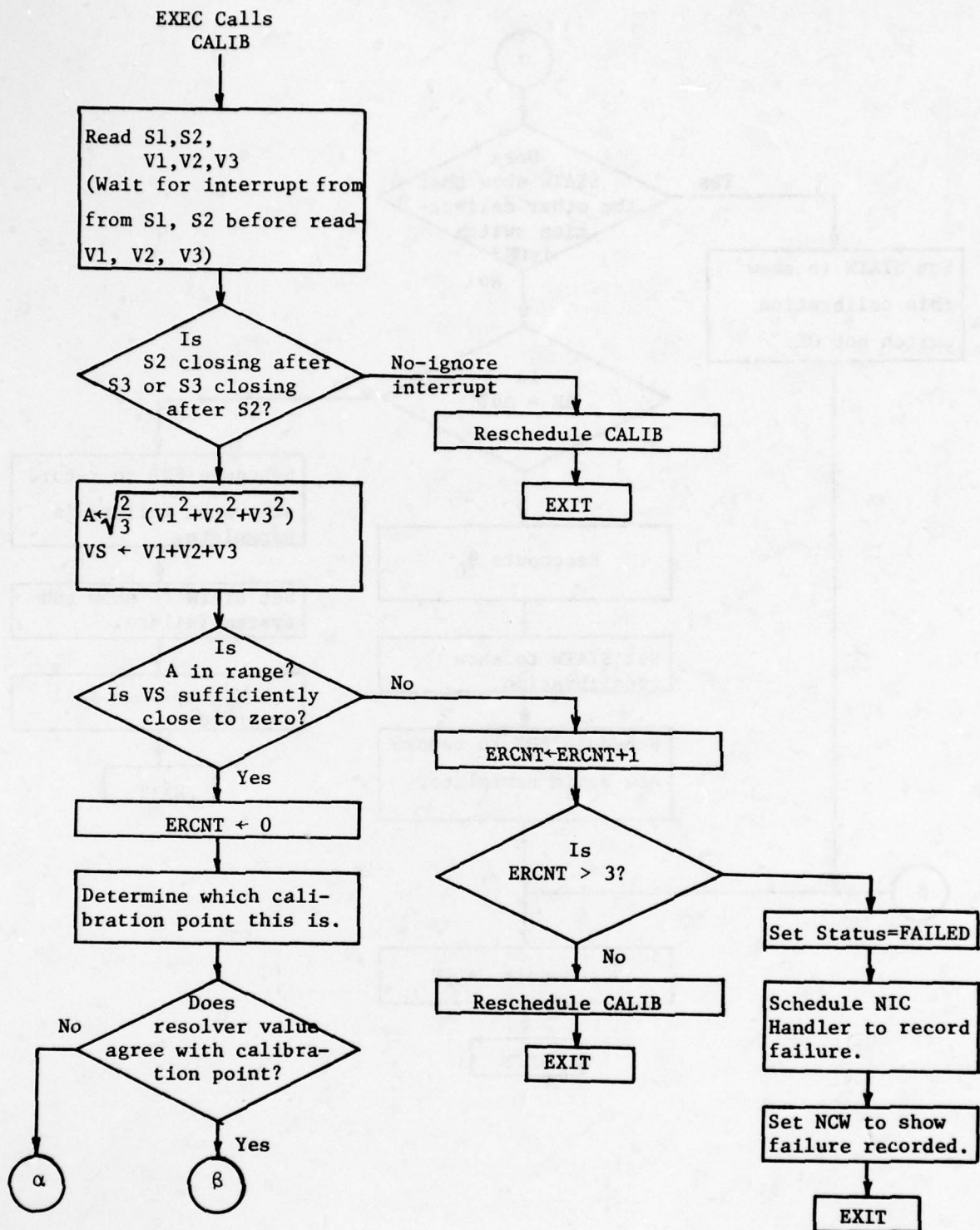


Figure 55 Task CALIB (Sheet 1 of 2)

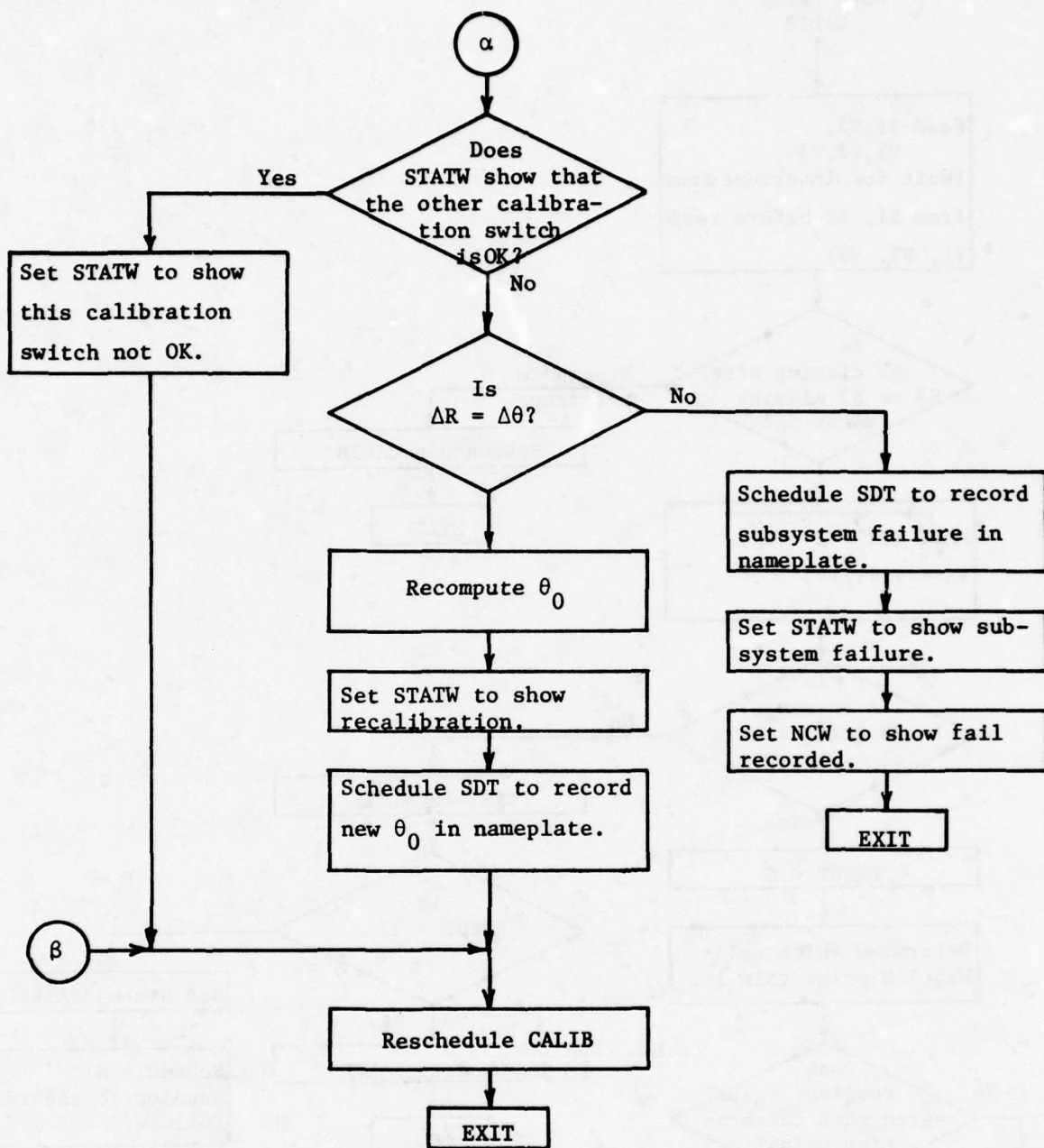


Figure 55 (Continued - Sheet 2 of 2)

Task to store failure and calibration reports in the nameplate. A flowchart for this task is shown in Figure 55 . Note that V1, V2, V3 must be read immediately after the ICA Group B interrupt.

7.8.3.3 Task LIMIT

This task waits for one of the switches S1 or S4 to close, then it reports the closure to LMG and it terminates, thereby becoming inactive until rescheduled by LMG/DAIS command. A flowchart for this task appears in Figure 56 .

7.8.3.4 Subsystem Diagnostic Task

This is the standard LM subsystem diagnostic task. When scheduled by the Configuration Task, it calls subroutine INIT which sets ERCNT (the error counter), STATW (the status word), and NCW (the nameplate control word) to zero, then schedules CALIB (MEAS and LIMIT are scheduled by LMG/DAIS). A flowchart is shown in Figure 57 .

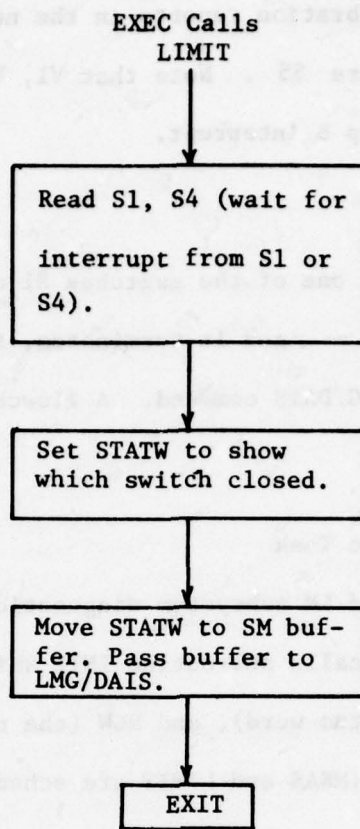


Figure 56 Task LIMIT

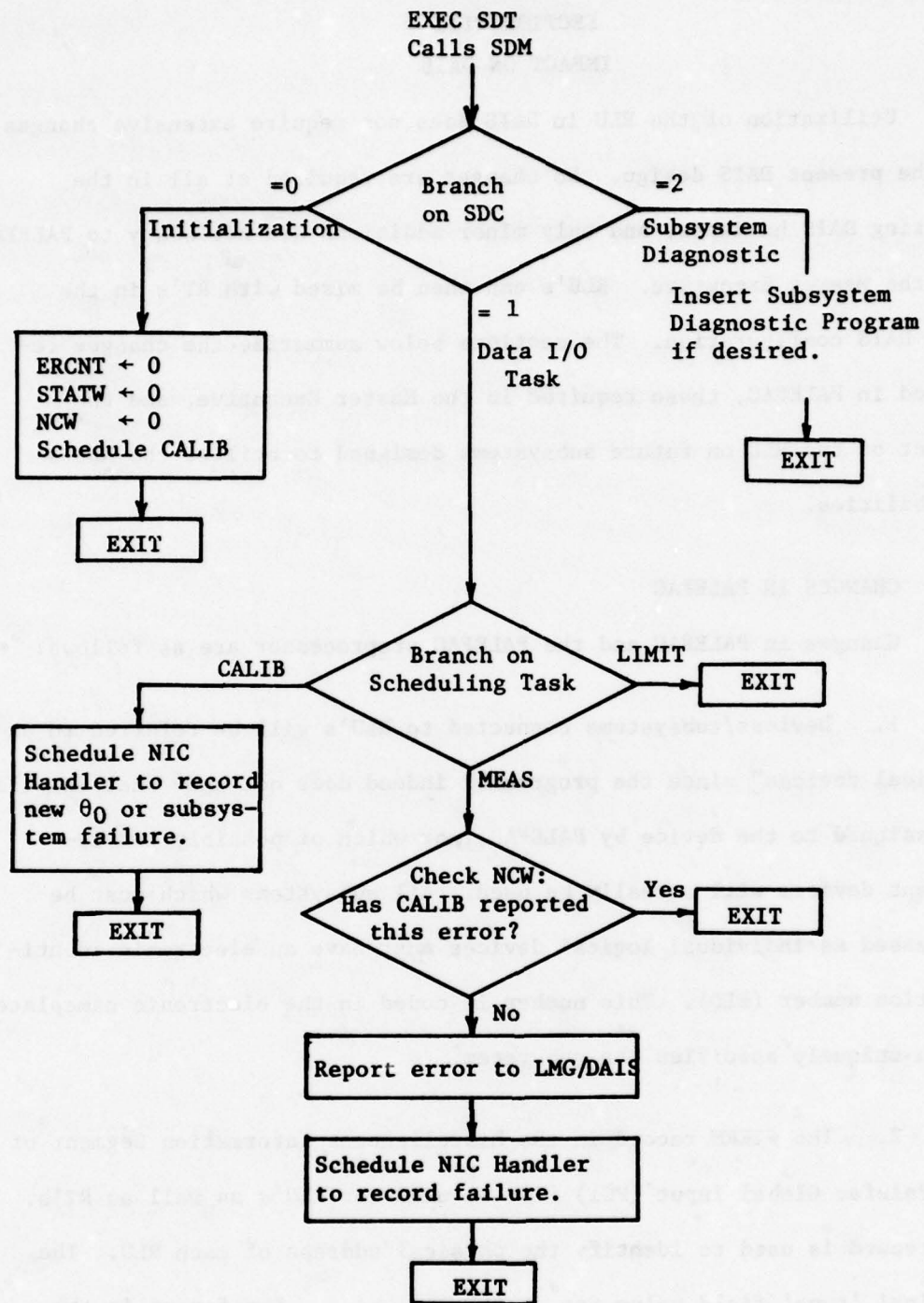


Figure 57 Subsystem Diagnostic Module

SECTION VIII

IMPACT ON DAIS

Utilization of the RLU in DAIS does not require extensive changes to the present DAIS design. No changes are required at all in the existing DAIS hardware, and only minor additions are necessary to PALEFAC and the Master Executive. RLU's can then be mixed with RT's in the same DAIS configuration. The sections below summarize the changes required in PALEFAC, those required in the Master Executive, and the impact of the RLU on future subsystems designed to utilize the RLU's capabilities.

8.1 CHANGES IN PALEFAC

Changes in PALEFAC and the PALEFAC preprocessor are as follows:

1. Devices/subsystems connected to RLU's will be referred to as "logical devices" since the programmer indeed does not know what SA will be assigned to the device by PALEFAC, nor which of possibly two redundant devices will actually be used. All subsystems which must be addressed as individual logical devices must have an electronic identification number (EID). This number is coded in the electronic nameplate which uniquely specifies the subsystem.

2. The #TERM record in the Miscellaneous Information Segment of the Palefac Global Input (PGI) file is used for RLU's as well as RT's. One record is used to identify the physical address of each RLU. The terminal 'type' field value for an RLU will be 4. The format is the same as at present in DAIS:

#TERM terminal-number type poll

terminal-number = multiplex terminal address (0-30)

type = type of terminal (0-7)

0: BCIU - Master processor

1: BCIU - Remote processor

2: Remote Terminal (RT)

3: Station Logic Unit

4: Remote Link Unit (RLU)

5-7: Non Standard Terminals

poll = capability to present asynchronous transmission
requests (0-1)

0 = no poll

1 = poll

Example: #TERM 16 4 1

3. A new #LOGICAL record is required to identify each logical
device in the system. The format of the record is:

#LOGICAL symbolic-name eid asynch rlu

symbolic-name = arbitrary name assigned to subsystem
by programmer.

eid = electronic ID # of the subsystem.

asynch = capability to present asynchronous transmission
requests (0-1).

0 = not asynchronous.

1 = asynchronous (PALEFAC will assign an activity

register value).

rlu = the RLU address (TA) for this device.

Example: #LOGICAL RADAR 12345678 0 23

4. In the Bus Message Segment of the PGI file, the source and destination of a bus message are normally specified in the FROM (data area) and TO (data area) fields of the bus message records. When the message is to or from a logical device, "data area" will be the symbolic-name of the subsystem as specified in the #LOGICAL record in the Miscellaneous Information Segment.

Example: FROM (RADAR) TO (IC12)

5. The ACTREG field of a PGI bus message record is not needed for asynchronous transmissions from an RLU. Palefac will choose an activity register pattern to be assigned to each asynchronous logical device as indicated by its #LOGICAL record.

6. PALEFAC assigns subaddresses as required for every subsystem and puts together the bus message tables.

7. As previously detailed, PALEFAC constructs the Terminal Configuration Table (TCT) and the Device Configuration Table (DCT). (See Section 2.2.1)

8.2 CHANGES IN THE MASTER EXECUTIVE

Changes required in the Master Executive are summarized as follows:

1. At startup initialization, the Master Executive must inform each RLU of what subaddresses have been assigned

to each subsystem. In addition each asynchronous device is told what to use for the activity register in response to mode command 16. (See Section 2.2.2).

2. Error/failure management is modified as described in Section 2.3.

8.3 CHANGES IN THE SUBSYSTEMS

The changes required in DAIS subsystems for operation with the RLU are summarized as follows:

1. Each subsystem interfaced to DAIS will require an electronic identification number and the specification of parameters to select the appropriate Link Module's interface configuration.

2. Cables to subsystems will have to be modified to accept the standardized connector which includes both universal interface signals as well as the subsystem information channel for communication with the nameplate.

3. All subsystems with serial interfaces will require a modification of the handshake logic to accommodate the link module's serial interface.

4. The contents of the nameplate for each subsystem must be specified and the nameplate configured and installed on each subsystem.

5. The Link Module will support all presently available interface signal types except output contacts, facility interface and pulse counter/

torques. These interfaces cannot be implemented with the standard link module and require individual implementation as special purpose link modules.

The following changes are desirable but are not essential to the RLU implementation:

7. Hardware signals and data tables which may be used to stimulate the subsystem for the purpose of testing should be incorporated as part of the subsystem design. Control of the testing conditions should be made available to the link module for the purpose of allowing automatic subsystem diagnostic testing.

8. Error analysis programs with the capability of performing logical decisions when errors are detected should be added to enhance the error recovery capability of a link module.

9. Data conversion, diagnostic and error analysis programs should be incorporated in the nameplate. Information related to subsystems malfunctions and maintenance should be stored as retrievable records in the electronic nameplate.

APPENDIX A

THE REMOTE LINK UNIT-AN ADVANCED REMOTE TERMINAL CONCEPT

Reprinted from

IEEE Transactions on Industrial Electronics
and Control Instrumentation

August 1979

The Remote Link Unit—An Advanced Remote Terminal Concept

CARLOS J. TAVORA, SENIOR MEMBER, IEEE

Abstract—Remote terminal units are extensively used in digital automation systems to interface a central processing unit (CPU) to remotely located sensors and actuators. This paper presents a conceptual description of the remote link unit (RLU) which overcomes inherent limitations of remote terminal units. The RLU utilizes a single type of interface card to support most process I/O interfaces. The RLU identifies sensors, actuators, and subsystems through the use of electronic nameplates. Each nameplate provides information regarding device function, location, interface parameters, and calibration status. In addition it stores programs for device handling, engineering unit conversion, and device diagnosis. The RLU allows sensors and actuators to be relocated or substituted without requiring changes in the CPU software to correct device addresses or conversion constants. The electronic nameplate may also be used for inventory control, automatic calibration, and maintenance of devices.

I. INTRODUCTION

REMOTE terminal units (RTU) are used extensively by designers of digital data acquisition and control systems to interface the central processing unit (CPU) with devices which are not close enough for connection to the I/O bus [1]–[7]. The remote terminal, as the name implies, provides termination for computer signals at a remote location through specialized termination cards (interface modules). Communication between the CPU and a remote terminal is implemented through serial digital transmission [8]–[10]. A controller card at the remote terminal carries out all functions required to support the operation of interface modules. These functions include message reception and transmission, serial to parallel format conversion, communication error detection, message interpretation, and timing and control of data transfer to and from interface modules. When interfaced through a remote terminal, a device is identified by the subaddress location of the interface module and the channel through which it is connected. This arrangement requires the use of address tables associating subsystems with their corresponding interface modules and channels. A meticulous and precise address table modification procedure must therefore be carried out whenever a change in system configuration takes place.

The remote terminal illustrated in Fig. 1, supports CPU interfaces to a thermocouple, a heater control, and a display

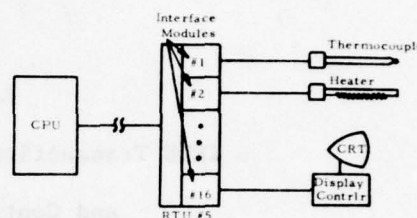


Fig. 1. Remote terminal interfaces to a thermocouple, a heater control, and a display subsystem.

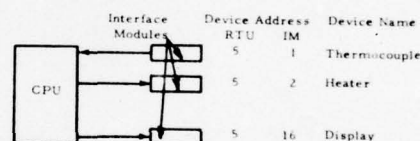


Fig. 2. The thermocouple, heater control, and display subsystem are seen by the CPU as interface modules of a remote terminal.

generator. Typically, one interface module is required for each interfaced device. The address configuration of these subsystems as seen by the CPU is depicted in Fig. 2. Data transfers to or from each device are accomplished with commands which have distinct formats and which reference different addresses. If, for example, the interface modules used with the thermocouple (A/D card) and the heater control (D/A card) are interchanged or relocated to another remote terminal, the CPU will not be able to communicate with either device. One may conclude that remote terminal interfaces require perfect coordination between the assignment of devices to interface modules and the corresponding entries in the device address table. This requirement is a nuisance to the technical personnel who must carry out tests utilizing digital data acquisition and control systems to perform automated test on test articles requiring a large number of sensors and actuators. Each time reconfiguration must take place either as a result of a test modification, or to correct the malfunctioning of a sensor or interface device, modifications must be concurrently made on the software device address tables to reflect the changes in system configuration. This not only requires careful planning and data base manipulation, but involves a considerable amount of documentation which is time consuming and is susceptible to human error.

An additional limitation of a typical remote terminal is the requirement that it must support several different types of interface modules [3]–[6]. Analog input and output, serial or

Manuscript received December 18, 1978; revised April 26, 1979. This work was supported initially by the University of Houston under Grant NROP-EE-K-91. A study of the feasibility of implementation in avionic systems is being supported by the U.S. Air Force Systems Command, Aeronautical Systems Division under Contract F33615-78-C-1634.

The author is with the Electrical Engineering Department, University of Houston, Houston, TX 77004.

TABLE I
FEATURES OF THE REMOTE LINK UNIT (RLU)

1. AUTOMATIC IDENTIFICATION OF INTERFACED INSTRUMENTATION
• RETRIEVES INFORMATION FROM THE ELECTRONIC NAME-PLATE OF AN INTERFACED DEVICE
• MAINTAINS A DIRECTORY OF INTERFACED DEVICES WITH NAME DESCRIPTOR AND LOGICAL UNIT NUMBER
• IDENTIFIES LOCATION AND FUNCTION OF EACH DEVICE WITHIN A SYSTEM
2. DEVICE INDEPENDENT I/O INTERFACES
• CONVERTS THE DATA FROM EACH DEVICE INTO A CONVENIENT DEVICE INDEPENDENT FORMAT
• ALLOWS THE CPU TO ADDRESS DEVICES BY EITHER DEVICE IDENTIFIER OR LOGICAL UNIT NUMBER
3. UNIVERSAL INTERFACE MODULES
• ADAPTS THE INTERFACE SIGNALS TO SATISFY THE DEVICE SIGNAL REQUIREMENTS (LEVELS & TIMING)
• USES A STANDARD CONNECTOR FOR ALL DEVICE INTERFACES
• UPLOADS SPECIAL PROGRAMS (DIAGNOSTICS, DATA CONVERSION, AND DATA VALIDITY CHECK)
4. PROCESSING AT EACH INTERFACE MODULE
• PERFORMS SYSTEM LEVEL CRITICAL CONTROL FUNCTIONS IN CASE OF SYSTEM FRAGMENTATION
• CONVERSION OF DATA FORMAT AND DATA VALIDITY CHECK
• PERIODIC TESTING AND CALIBRATION OF INTERFACED DEVICES

parallel digital input and output, and many other interface signal formats are required to handle devices which might be connected through a remote terminal. Indeed, even within a single category such as serial digital input there can be many subcategories necessitating still more distinct interface types. The inventory and maintenance expertise necessary to support the large variety of interface modules required by distinct external devices poses a major logistics problem.

Still another limitation of the remote terminal is its lack of ability to operate stand-alone supporting control and monitoring function in a degraded mode without CPU assistance. Such capability is essential for operation critical processes which require well-defined shutdown sequences in order to avoid catastrophic losses in production or equipment as a result of system control fragmentation.

The problems associated with remote terminals can be traced to a basic operational concept which treats them as peripheral devices rather than as a transparent means for communication between CPU and devices. An objective review of the desirable features which should be provided by a general purpose remote interface between CPU and external devices, has led to the concept of a remote link unit (RLU) which is proposed in this paper. A summary of the significant features of the RLU is outlined in Table I.

The RLU provides a complete interface since, in addition to implementing data transfers, timing and control signals, it will also provide a channel through which the CPU can interrogate each external device for that device's identity. The RLU interface modules are transparent in the sense that the CPU addresses devices with which it desires to communicate using logical device numbers assigned during run time. This function may be accomplished by maintaining a dynamically

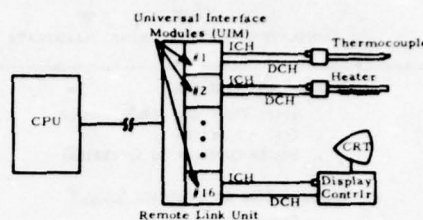


Fig. 3. Remote link interfaces to a thermocouple, a heater control, and a display subsystem.

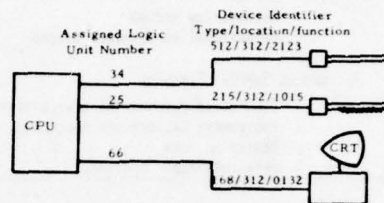


Fig. 4. The thermocouple, heater control, and display subsystem are seen by the CPU as independent peripheral devices. Each device has a unique identifier and a CPU assigned logical unit number through which it is addressed.

updated subsystem directory. When a command is received, the RLU checks its directory and proceeds to transfer the information to the corresponding subsystem. Since this directory is dynamically updated by the RLU to reflect the current external device configuration, it is possible to modify that configuration, e.g., move a device connection from one interface module to another, without modifying the CPU device tables. An illustration of the RLU interface to a thermocouple, a heater control, and a display generator is presented in Fig. 3. In this case these devices are seen by the CPU as individual and peripheral, each addressable as a logical device as shown in Fig. 4.

The RLU will support universal interface modules (UIM) for connection to external devices. Such interface modules will provide all types of signal interfaces (ac or dc analog, serial or parallel digital) on a single card which can be configured on demand to accommodate any device interface signal requirements. To accomplish this each UIM should have, in addition to the conventional data channel (DCH), a separate identification channel (ICH) for subsystem identification and interface requirements specifications as shown in Fig. 3. This channel should have an identical format for all devices. A communication protocol for the identification channel should be used to control the transfer of information between UIM and device.

The RLU should provide processing at the UIM thus allowing several functions to be performed at the remote location independent of CPU operation. This feature allows validity checks to be carried out on data originating at external devices prior to its transfer to the CPU. In addition, the proper operation of a subsystem may be monitored through its status and data. Format conversion and data limit checking may also be performed at the interface and thus reduce the CPU processing load. By allowing interface modules to have stand-alone processing capability, control and display functions may be executed at the RLU in case of system fragmentation. The re-

TABLE II
CONTENTS OF THE ELECTRONIC NAMEPLATE

1. DEVICE IDENTIFICATION
. TYPE, MODEL AND SERIAL NUMBER
. SYSTEM FUNCTION
. SYSTEM COORDINATES (LOCATION)
2. SPECIFICATION OF INTERFACE SIGNALS
. SIGNAL LEVELS
. HANDSHAKE AND TIMING
. TRANSMISSION RATE
3. DEVICE OPERATIONAL RECORD
. CALIBRATION RECORD
. OPERATIONAL PERFORMANCE RECORD
4. DEVICE SUPPORT FIRMWARE
. DIAGNOSTIC PROGRAM FOR DEVICE TESTING
. INSTRUMENT CALIBRATION PROCEDURE
. DEVICE HANDLER
. DATA CONVERSION ROUTINES

remote link unit described in this paper is based on three major concepts which will be described next: the electronic nameplate, the universal interface module, and the link manager.

II. THE ELECTRONIC NAMEPLATE

The electronic nameplate is a key concept which not only leads to the implementation of the remote link unit but also supports the automatic calibration, testing and inventorying of devices. As the name indicates, the electronic nameplate provides device identification by supplying type, model, and serial number. In this sense, the electronic nameplate supplies information similar to that provided by the universal product code (UPC) in packaging. However, the electronic nameplate will provide additional information which far exceeds the simple identification of a device. This information will include the following: device function and location as a part of a system, parameters specifying the type of interface signals required, records of the operational performance and calibration of the device, and, finally, programs (subroutines, routines, and tasks) which may be used to perform device diagnostic testing, device handling and data conversion.

By storing in each instrument the software required to convert its data into a device independent format, it is possible to design system programs that are device independent. Henceforth, similar instruments having distinct interface characteristics may be interchanged without effecting the CPU software. A summary of the possible contents of the electronic nameplate is presented in Table II.

A possible architecture for the electronic nameplate is illustrated in Fig. 5. An extension port on the device nameplate is used to interface a secondary nameplate. This port is used to retrieve information related to the device's function and location as part of a system from a nameplate installed in the system's frame where the device is mounted. In addition to the interface port, the electronic nameplate should include a read-only memory containing the device identification and interface parameters, as well as device handlers and other programs which may be retrieved by the RLU. An electrically

160

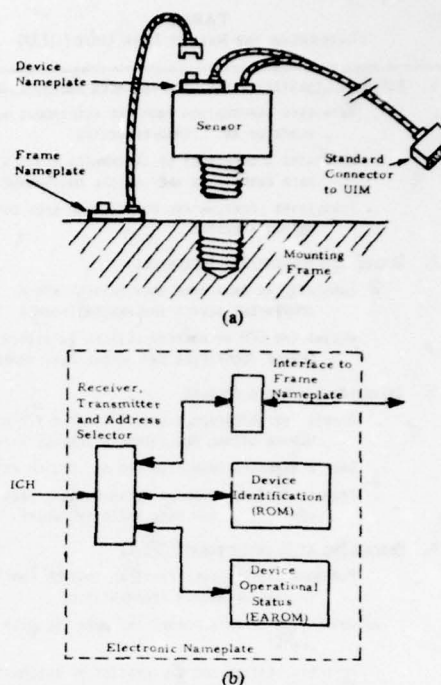


Fig. 5. The electronic nameplate is distributed among the device and the mounting frame (a). The internal architecture of the nameplate includes a controller, two interfaces, and memory (b).

alterable memory should be provided for storing the device calibration and operation records. Transfer of information from the electronic nameplate to the universal interface module may be implemented with a serial data communication controller powered by the interface module.

The electronic nameplate may find extensive use in the automatic calibration, maintenance and inventory control of external system devices.

III. THE UNIVERSAL INTERFACE MODULE

The universal interface module (UIM) will support an interface for any device which can be configured with the assistance of an electronic nameplate. The interface module connector will have a preestablished pin assignment which may support several distinct functions through the same set of pins. The exact configuration of the pins in terms of data signals and control signals is specified from the configuration parameters obtained from the electronic nameplate. Initially, all lines of the universal interface connector are maintained at a high-Z state. The universal interface module, upon detecting the connection to an external device will proceed to interrogate the electronic nameplate and retrieve its data. The device identification parameters will be the first retrieved. These will be followed by the interface requirement parameters which are used to select the appropriate data and control signals to the interface connector. The electronic nameplate will be interrogated for the existence of a diagnostic program. This program should be uploaded to the interface module for execution to establish the operational status of the interface module and

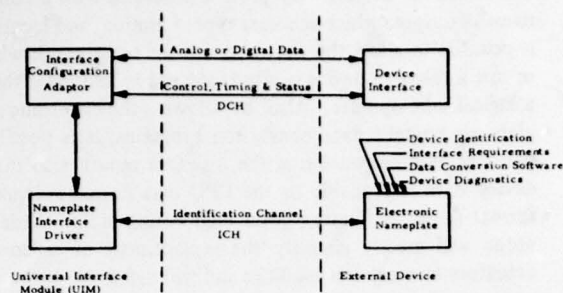


Fig. 6. Block diagram of the interface between a universal interface module and an external device.

TABLE III
CHARACTERISTICS OF THE UNIVERSAL INTERFACE MODULE (UIM)

1. STANDARD SENSOR/ACTUATOR INTERFACE	
•	UTILIZES A STANDARD CONNECTOR
•	SUPPORTS INTERFACE TO THE ELECTRONIC NAMEPLATE (IDENTIFICATION CHANNEL)
•	ADAPTS THE INTERFACE TO SATISFY SENSOR/ACTUATOR SIGNALS (DATA CHANNEL)
2. IDENTIFICATION CHANNEL	
•	DETECTS DEVICE PRESENCE
•	IDENTIFIES INTERFACED DEVICE
•	MAINTAINS DEVICE OPERATIONAL RECORD
•	RETRIEVES PROGRAMS FROM THE NAMEPLATE
3. MODULE PROCESSING	
•	DEVICE INDEPENDENT PROCESSING
•	INITIALIZATION
•	PERIODIC SCAN
•	MODULE DIAGNOSTIC
•	DEVICE DEPENDENT PROCESSING
•	HIGH LEVEL INTERPRETIVE LANGUAGE
•	DATA CONVERSION
•	DEVICE DIAGNOSTIC
4. INTERFACE TO LINK MANAGER	
•	DEVICE INDEPENDENT FORMAT
•	DEVICE STATUS INFORMATION (DIRECTORY)
•	MODULE HARDWARE CONTROL AND STATUS
•	MODULE TASK CONTROL AND STATUS
•	INTERMODULE AND CPU COMMUNICATION

the device. The retrieval of data conversion programs from the electronic nameplate into the interface module will allow data to be maintained in a device independent format. The interface elements of an UIM and a device are shown in Fig. 6.

In order to maximize the flexibility of the universal interface module, a general purpose interpretive language should be used at the interface module for processing data and controlling the device. The use of an interpretive language will make the device programs processor independent. An interpretive language such as APL or Basic may be used as such a machine independent language. The device data processed by the interface module is transferred to the CPU through the RLU's link manager which communicates with interface modules through shared memory. This shared memory may also be used to transfer processing commands to the interface modules, store the operational status of the device, and maintain all information pertinent to the RLU device directory such as device type, function, location, and logical unit number. A summary of the features provided by the universal interface module is presented in Table III.

IV. LINK MANAGER

The link manager provides the interface between the CPU and universal interface modules. The functions performed by the link manager include the upkeep of the directory of interfaced devices, the control of data flow between the CPU and universal interface modules, the control of the RLU hardware resources, and the coordination of local processing among the universal interface modules to support stand-alone RLU operation.

The directory of interfaced devices contains the names of all devices which are interfaced through the RLU. The directory contains the location of the universal interface module through which the device is interfaced and possesses a logical unit number assigned to the device by the CPU. In this manner, a cross reference between physical location of an interface module, the name of the device it interfaces, and the corresponding logical unit number is established. In addition to the identification, the directory provides information relevant to the operational status of each device and its corresponding universal interface module. This information should be generated through the execution of diagnostic programs which exercise the interface module and the external device. The RLU directory should also provide identification and execution status information on all external programs loaded into an interface module. These programs may have been uploaded from a device to the universal interface module or downloaded from the CPU. Whenever the system configuration is modified either to correct a device failure or to alter the scope of the system functions, an update of the system directories takes place. When a change of configuration occurs, all affected remote link units update their directories and the CPU is informed that it should update its copy of the directory tables so as to reflect the correct system configuration. In normal operation, the link manager updates the directory of interfaced devices based on the data provided by the universal interface modules. When the system is initialized, the CPU should interrogate each link manager in order to obtain the name descriptor of all interfaced devices, following which the CPU issues a logical device number to each device for addressing during normal system operation. The link manager identifies the address tag (logical device number) of data from the CPU and routes it to the corresponding universal interface module. In this sense the link manager performs the same functions as a front-end processor used as a data concentrator for a multiterminal environment.

The link manager controls the hardware resources of the RLU such as redundant power supplies, alternate communication channels, and available memory in order to maintain its operation in the eventuality of failure of any one of these components. As such, it can control these resources based on its internal status or as a result of commands received from the CPU. The link manager will monitor the system communication to decide when it is isolated from the CPU, in which case it will coordinate intermodule processing for stand-alone operation. This will allow critical control functions to be processed locally such as an orderly shutdown of devices and functions which cannot operate when the remaining system is isolated. A summary of the functions provided by the link manager is

TABLE IV
FUNCTIONS OF THE LINK MANAGER

1. MAINTENANCE OF RLU DIRECTORY
. DEVICES (NAME, FUNCTION, LOCATION, STATUS)
. MODULES (HARDWARE STATUS, SOFTWARE STATUS)
. RLU (COMMUNICATION STATUS, PROCESSING FUNCTIONS, HARDWARE STATUS)
2. COMMUNICATION TRAFFIC CONTROL
. TRANSFER OF DATA AND COMMANDS BETWEEN CPU AND UIM'S
. COMMUNICATION ERROR RECOVERY
. DETECTION OF RLU ISOLATION FROM SYSTEM
3. COORDINATION OF RLU PROCESSING
. CONTROL OF SYSTEM LEVEL TASK PROCESSING AMONG UIMs (RLU STAND-ALONE OPERATION)
. MANAGEMENT OF MULTIPLE REDUNDANT SENSORS
. SYSTEM POWER-UP AND SHUT-DOWN SEQUENCES
4. CONTROL OF RLU HARDWARE RESOURCES
. SELECTION OF RLU POWER SOURCE
. SELECTION OF COMMUNICATION BUS
. ISOLATION OF DEFECTIVE RLU MODULES

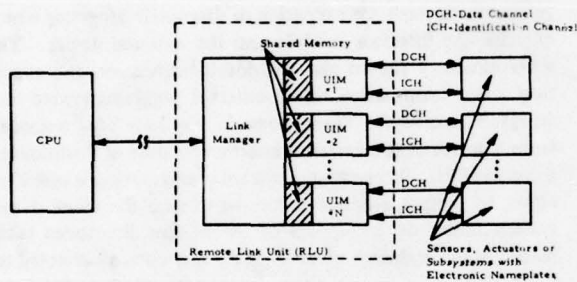


Fig. 7. Components and interfaces of the remote link unit.

outlined in Table IV. The architecture of the remote link unit in terms of the link manager and the universal interface modules is presented in Fig. 7.

V. CONCLUSIONS

This paper has presented the concept of a remote link unit which incorporates several enhancements not available in existing remote terminal units. The most significant concepts introduced are the electronic nameplate and the universal interface module. The electronic nameplate will significantly simplify the design of software for systems which must be re-configured dynamically, and will facilitate the checkout and

calibration of devices. By providing devices with a complete name descriptor which includes type, function, and location, it is possible to allow the central processor to establish whether or not a required device is interfaced and to address it through a logical unit number. Also, by allowing the electronic nameplate to contain data conversion programs, it is possible to perform data conversion at the interface modules so that the device data is available to the CPU in a device independent format (such as floating point engineering units). This technique will greatly simplify the replacement of sensors and actuators in most test facilities and will make the device transparent to the programmer.

The concept of a universal interface module capable of supporting all external device interfaces and utilizing a standard connector will uncomplicate the interfacing procedures and significantly decrease the inventory of cards required to support process control interfaces. The capability of local processing at the interface module with programs which are either uploaded from a device or downloaded from the CPU, provides both flexibility and fallback capability.

The concepts presented in this paper may be integrated to implement the remote link unit or may be individually added to existing remote terminal units in order to enhance their operation as system components.

ACKNOWLEDGMENT

The author wishes to express his gratitude to T. A. Jagen, Jr., for the helpful discussions and suggestions.

REFERENCES

- [1] D. Horelick and R. S. Larsen, "CAMAC: A modular standard," *IEEE Spectrum*, vol. 13, pp. 50-55, Apr., 1976.
- [2] MODAC, *Reference Manual*, Modular Computer Systems, Inc., 1975.
- [3] RTP-Process Interface Subsystems, *User Manual*, Computer Products, Inc.
- [4] TRW 2000 Remote Terminal Unit, *Reference Manual*, TRW Controls.
- [5] Series 5000 Remote, *Instruction Manual*, Harris Controls, 1978.
- [6] Input/Output Interface Subsystems Model 11XX, *Tech. Manual*, Modular Computer Systems, Inc., 1975.
- [7] C. W. Rose, E. Linn, and J. D. Schoeffler, "Distributed micro-computer data acquisition," *Instrumentation Technology*, vol. 20, pp. 55-61, Dec. 1975.
- [8] J. Washburn, "Communications interface primer-Part I," *Instruments and Control Systems*, pp. 43-48, Mar., 1978.
- [9] —, "Communications interface primer-Part II," *Instruments and Control Systems*, pp. 59-64, Apr., 1978.
- [10] J. D. Schoeffler, "Data highway structures and their effects on applications," *ISA Trans.*, vol. 17, pp. 3-12, Jan., 1978.

APPENDIX B

PRESENT DAIS REMOTE TERMINALS

CONTENTS

SECTION	PAGE
B.1 MESSAGE STRUCTURE	164
B.2 RT MESSAGE DECODING AND DATA ROUTING SCHEME B-2	164
B.3 SYNCHRONOUS AND ASYNCHRONOUS MESSAGES	170
B.4 DAIS SYSTEM INITIALIZATION	175
B.5 ERROR RECOVERY	177
B.6 TCU AND INTERFACE MODULES	177
B.7 INTERFACE MODULE TESTING	180

This section summarizes the significant features of the present remote terminals and describes the manner in which they operate as elements of the DAIS system.

B.1 MESSAGE STRUCTURE

Communication between DAIS processors and remote terminals is implemented through messages consisting of command, status and data words.

The word formats and message protocol used in the DAIS system are summarized in Figures B-1 and B-2.

B.2 RT MESSAGE DECODING AND DATA ROUTING SCHEME

The processing of a command by a remote terminal is illustrated in Figure B-3. The transmit/receive flag selects one out of two blocks of 256 bytes in an EROM to be indexed by the command subaddress. The subaddress field contains an index with value between 1 and 31 that specifies a pointer to a set of channels and their corresponding interface modules stored in the EROM. The number of channels configured at a subaddress is specified by the word count field of the command. Subaddress 0 is reserved for commands directed to the remote terminal processor and is referred to as a mode code. When a mode code is received, the word count field of the command is used to specify one of possible 32 RT directed commands.

When a command is followed by data (receive command), the data received is stored in a data buffer (2 bytes/word) and its destination (channel/interface module) is copied from EROM into a command buffer as illustrated in Figure B-4. When the number of words specified by the word count is received, a flag is set in the corresponding command buffer byte to signal end of message. If the command sequence and data reception is error free, the RT will proceed to route data from the data buffer to the appropriate interface module as specified by the command buffer.

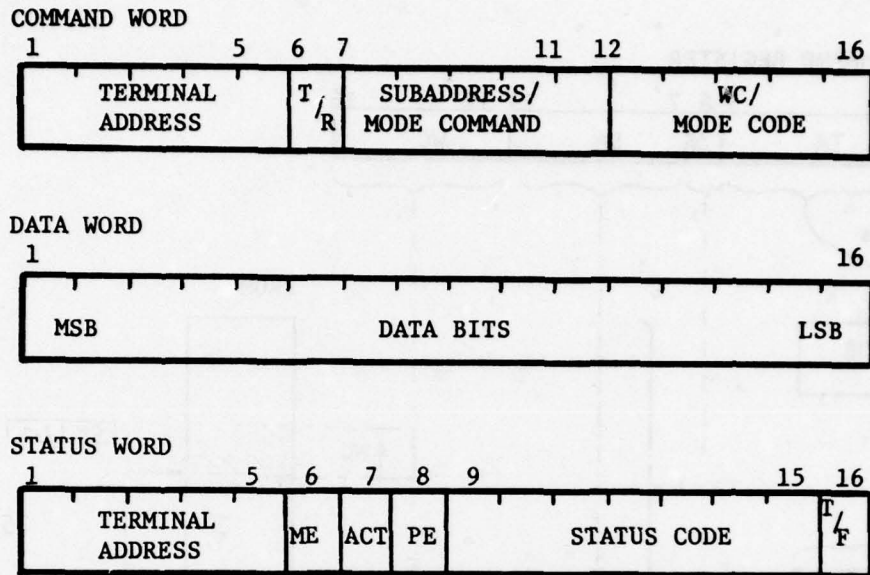


Figure B-1 DAIS Message Word Format

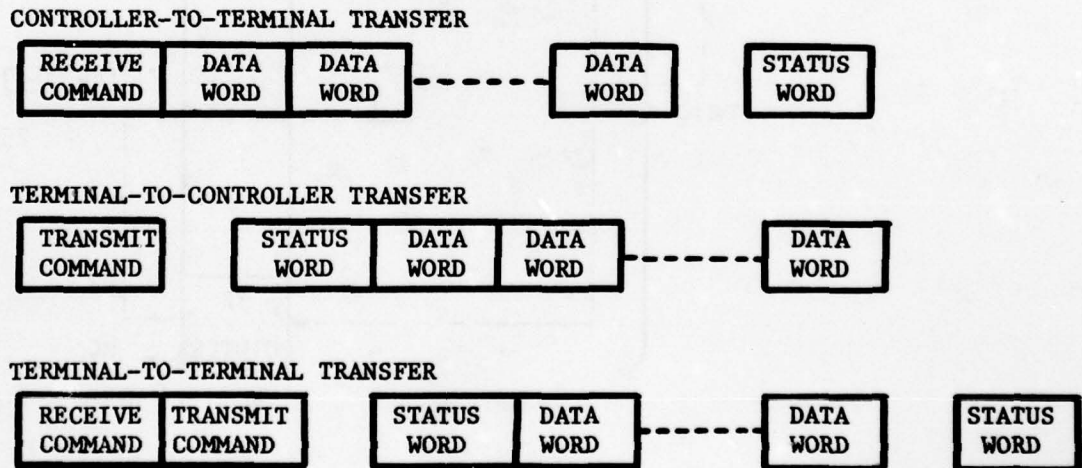


Figure B-2 DAIS Message Format

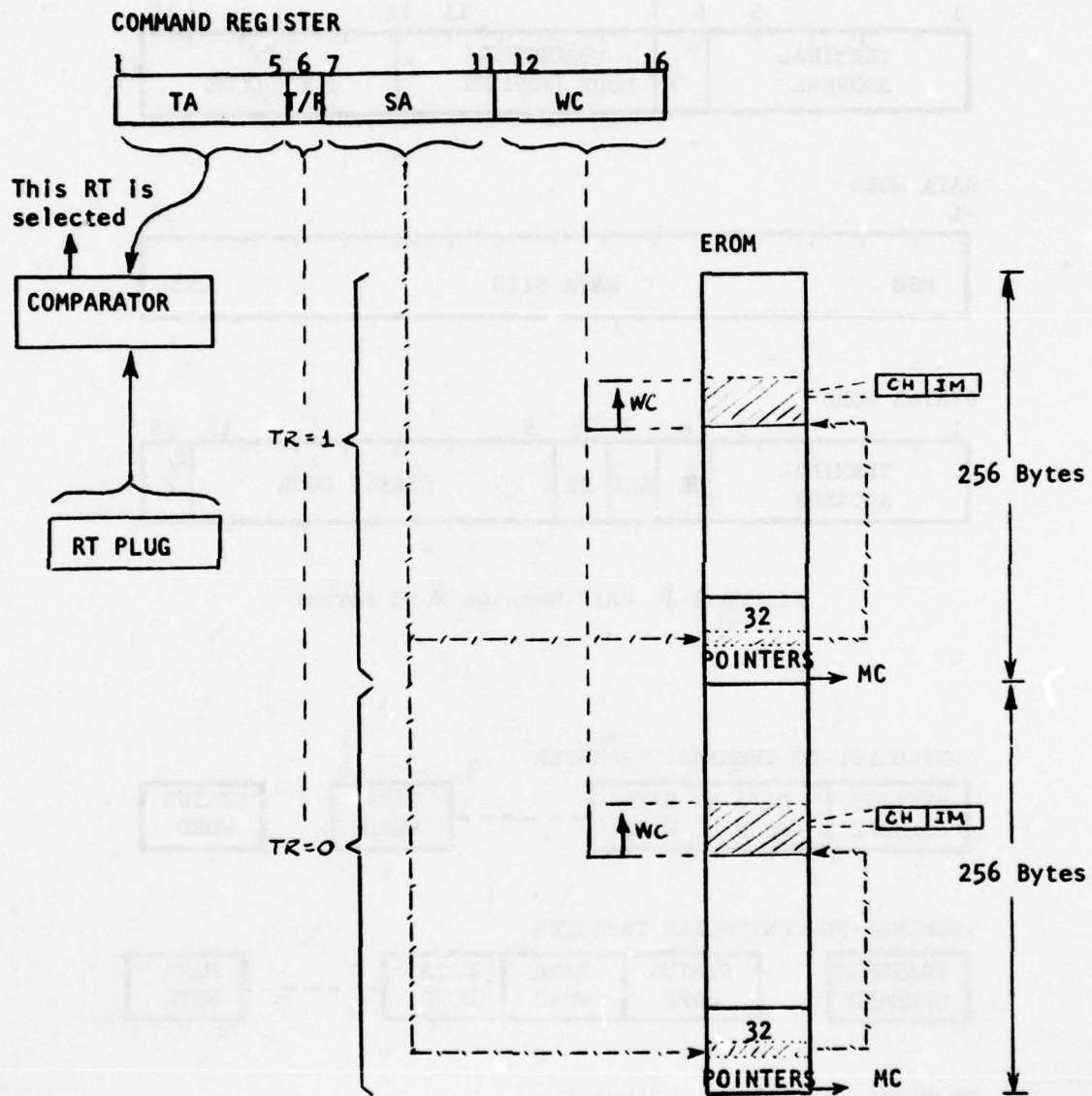


Figure B-3 RT command decoding.

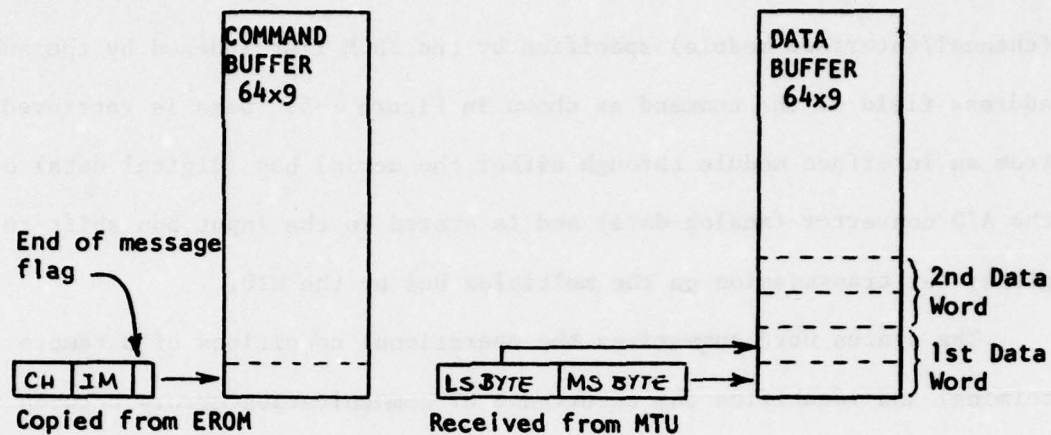


Figure B-4 Receive sequence.

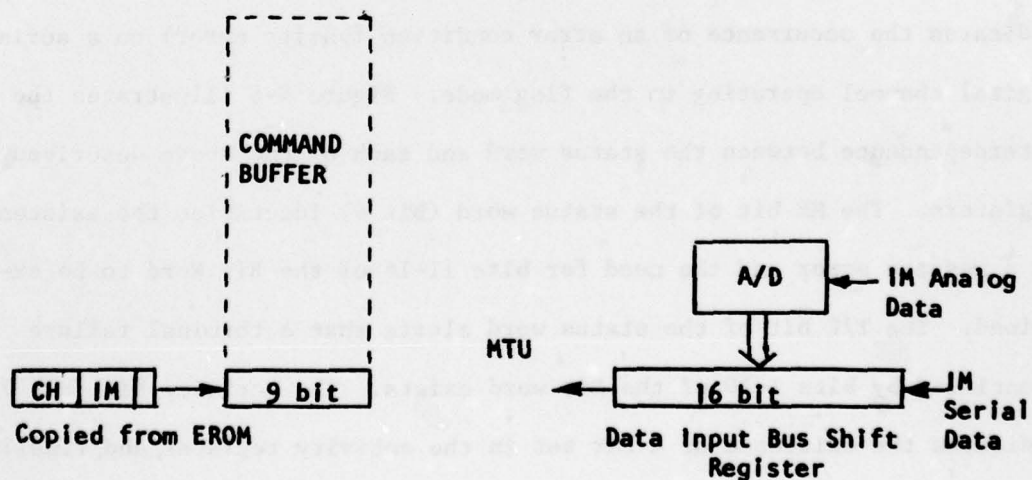


Figure B-5 Transmit Sequence

When a command specifies transmission of data by the remote terminal (transmit command), data words are fetched from the appropriate source (channel/interface module) specified by the EROM list indexed by the sub-address field of the command as shown in Figure B-5. Data is retrieved from an interface module through either the serial bus (digital data) or the A/D converter (analog data) and is stored in the input bus shift register for transmission on the multiplex bus by the MTU.

The status word summarizes the operational conditions of a remote terminal and identifies the occurrence of communication errors between the remote terminal and the DAIS processors. The status word summarizes the conditions of three registers: the Bit Word, the Activity Register and the Error Register. The Bit Word can be decomposed into two portions, bits 1-10 identify the self-test status of the terminal while bits 11-16 qualify the last message transfer. The activity register identifies the presence of data to be transferred from an asynchronous channel configured to operate in the flag mode to one of the DAIS processors. The error register indicates the occurrence of an error condition (parity error) on a serial digital channel operating in the flag mode. Figure B-6 illustrates the interdependence between the status word and each of the above described registers. The ME bit of the status word (bit 6) identifies the existence of a message error and the need for bits 11-16 of the Bit Word to be examined. The T/F bit of the status word alerts that a terminal failure identified by bits 1-10 of the Bit word exists. The activity bit (bit 7) indicates the existence of a bit set in the activity register, and, finally, the PE bit (bit 8) denotes the existence of a parity error in the error register. The relationship between bits in the error register and the

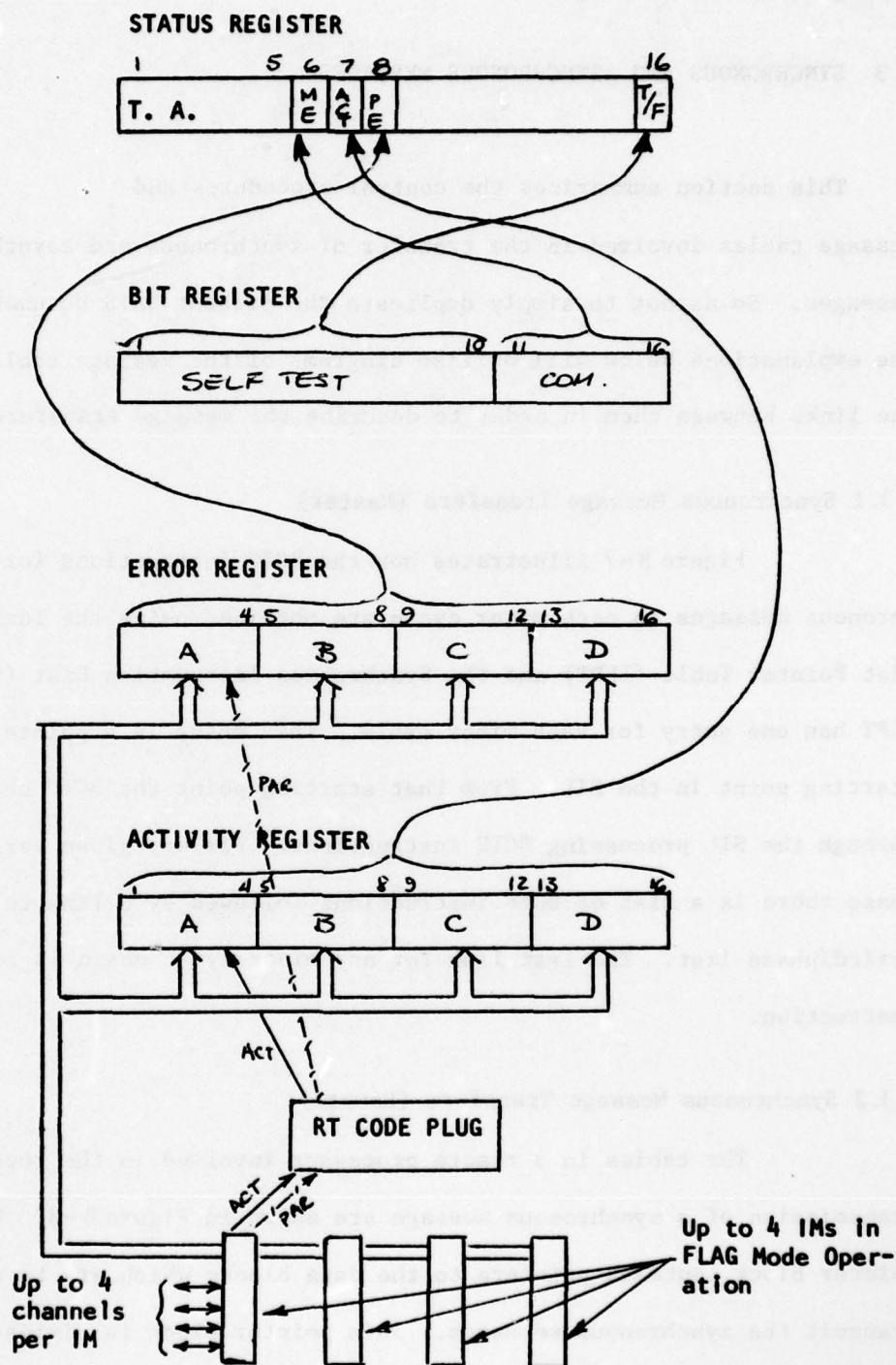


Figure B-6 Interaction between RT registers.

activity register and the corresponding interface modules is established through the RT Code plug.

B.3 SYNCHRONOUS AND ASYNCHRONOUS MESSAGES

This section summarizes the control procedures and message tables involved in the transfer of synchronous and asynchronous messages. So as not to simply duplicate the present DAIS documentation, the explanations below will utilize diagrams of the message tables and the links between them in order to describe the message transfers.

B.3.1 Synchronous Message Transfers (Master)

Figure B-7 illustrates how the BCIU instructions for the synchronous messages in each minor cycle are obtained using the Instruction List Pointer Table (ILPT) and the Synchronous Instruction List (SIL). The ILPT has one entry for each minor cycle. That entry is a pointer to the starting point in the SIL. From that starting point the BCIU chains through the SIL processing BCIU instructions. For any given period and phase there is a list of BCIU instructions followed by a link to the next period/phase list. The last link for any minor cycle chain is to a BCIU halt instruction.

B.3.2 Synchronous Message Transfers (Remote)

The tables in a remote processor involved in the reception or transmission of a synchronous message are shown in Figure B-8. The DMA pointer block contains pointers to the data blocks which are to receive or transmit the synchronous messages. This pointer block is divided into four sections: receive, even; transmit, even; receive, odd; transmit, odd.

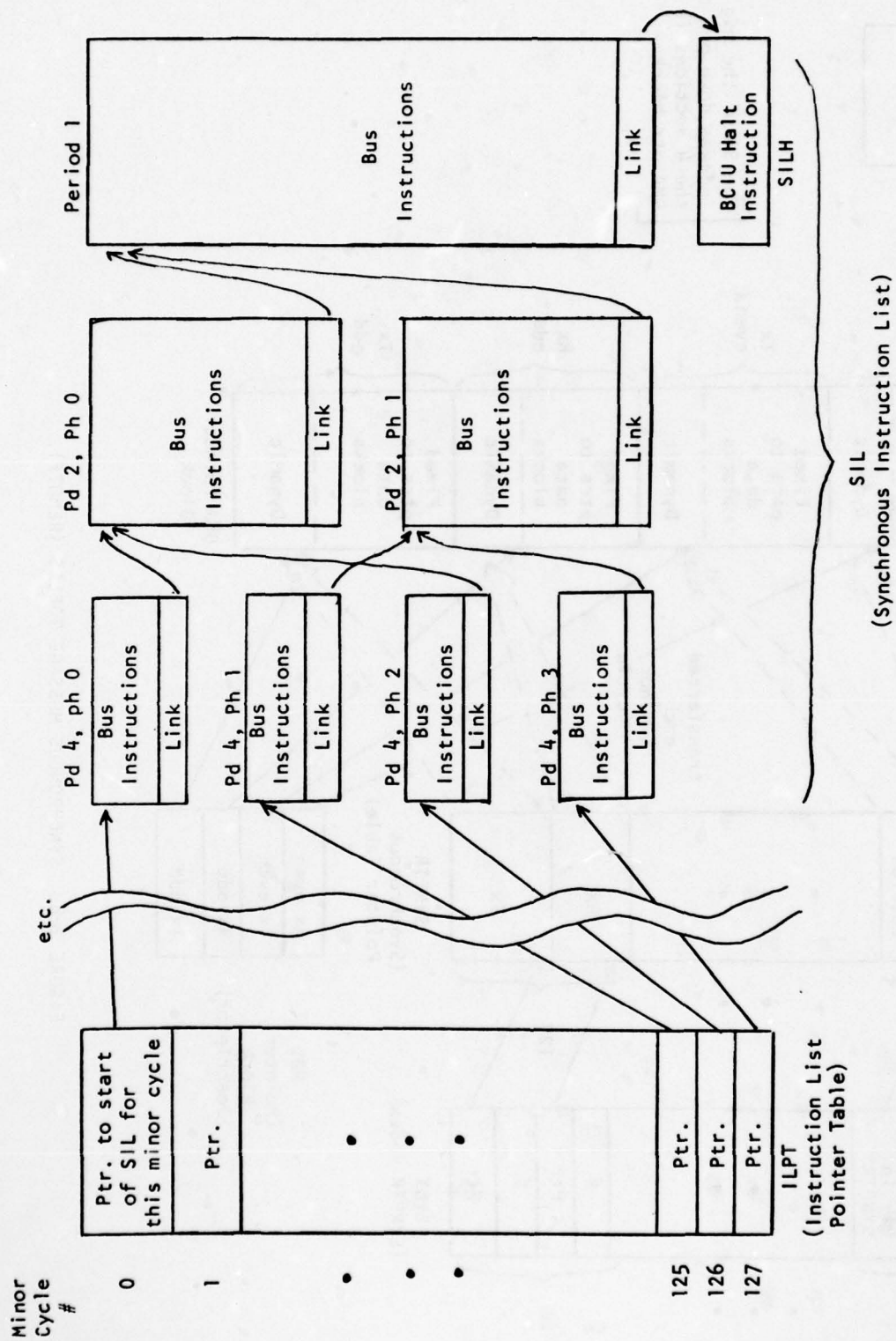


FIGURE B-7. SYNCHRONOUS MESSAGE TABLES (MASTER)

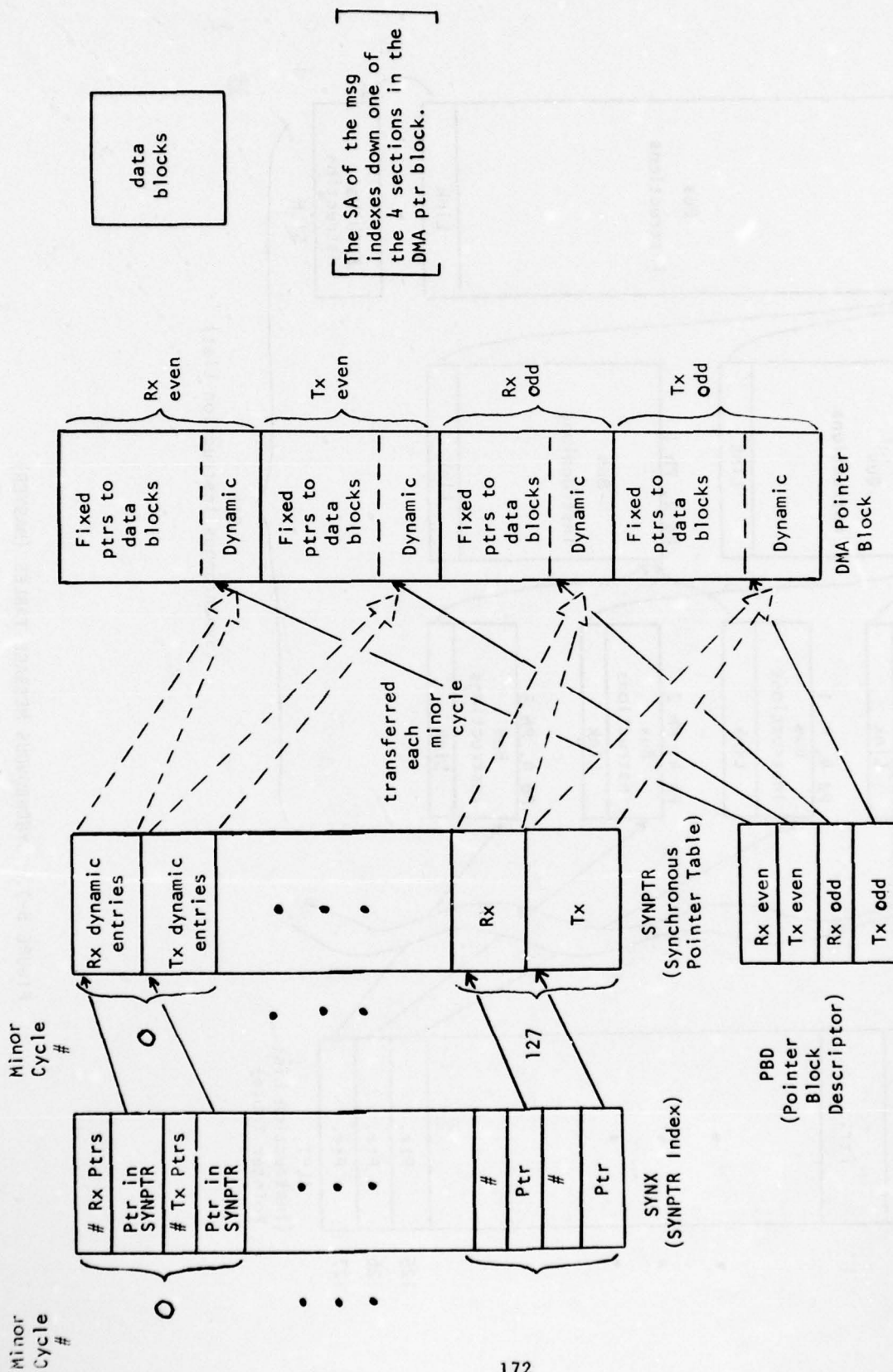


FIGURE B-8. SYNCHRONOUS MESSAGE TABLES (REMOTE)

Receive/transmit distinguishes between those messages being received and those to be transmitted. Odd/even distinguishes between odd and even minor cycle numbers. The subaddress (SA) in the message command word is used as an index down the appropriate one of these four lists to find the pointer to the data block in the compool area. The BCIU does all this automatically and performs the data transfer via direct memory access (DMA).

Each of the four sections of the DMA pointer block is actually further divided into fixed and dynamic subsections. The pointers in the fixed subsection are never changed. Those in the dynamic subsection can be changed each odd and even minor cycle. The pointer block descriptor (PBD) table contains four pointers to the top of each dynamic subsection.

For each minor cycle, entries in the SYNPTR Index (SYNX) table tell the local executive how many dynamic pointers are to be transferred and where to find them in the Synchronous Pointer (SYNPTR) Table. The appropriate entries in SYNPTR are then transferred to the corresponding dynamic subsections in the DMA Pointer Block.

B.3.3 Asynchronous Message Transfers (Master)

Figure B-9 shows the tables used by the master executive in processing a request from an RT for an asynchronous message transmission. When the BCIU interrupts the master processor due to a nonzero activity bit in a status word, the master executive sends mode code 16 to the RT to obtain the activity register. The RT number is then used to index down the Remote Asynchronous Table (RAT) to obtain a pointer into the Master Instruction Keys Table (MINK). RAT also gives the number of entries in MINK associated with that RT. Using the information in the activity register, the master executive finds which one of those entries in MINK is

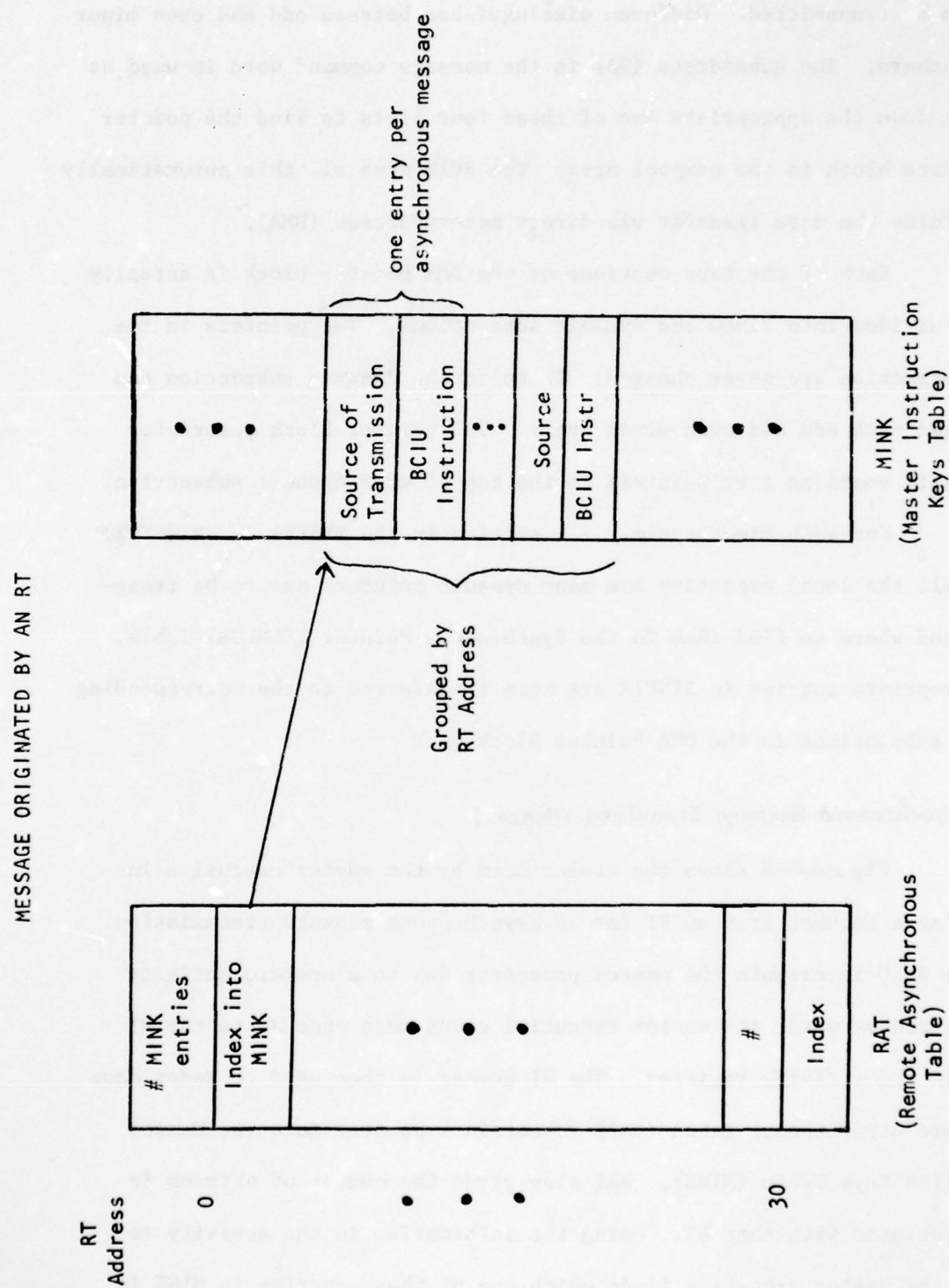


FIGURE B-9. ASYNCHRONOUS MESSAGE TABLES (MASTER)

AD-A080 126

HOUSTON UNIV TX DEPT OF ELECTRICAL ENGINEERING
REMOTE LINK UNIT FUNCTIONAL DESIGN: AN ADVANCED REMOTE TERMINAL--ETC(U)
OCT 79 C J TAVORA, J R GLOVER, G W BATTEN F33615-78-C-1634
AFAL -TR-79-1176 NL

UNCLASSIFIED

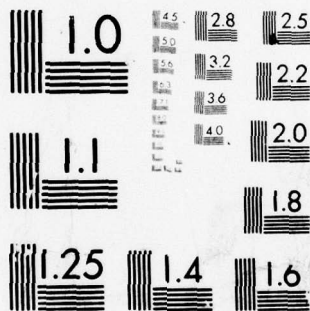
3 OF 3
AD
A080126



END
DATE
FILMED

*3-80

DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

associated with the requesting subsystem, and thus obtains the BCIU instructions necessary to effect the asynchronous message transfer.

B.3.4 Asynchronous Message Transfers (Remote)

Figure B-10 illustrates the message tables used by the local executive in processing a received asynchronous message. The first word of the message is the Last Command Register from the transmitting RT. Since the last command was a transmit command to that RT, it gives the RT/SA address of the subsystem from which the message is being received. The RT address is used to index down the Transmit Originate Address (TOAD) Table to find a pointer into the Subaddress Name Keys (SNAKE) Table, along with the number of entries in SNAKE for that RT. The local executive then finds the entry in SNAKE which corresponds to the subaddress obtained from the last command register. This entry is a pointer to the appropriate data block descriptor (DDB) in the Asynchronous DDB Area (ADDB). From the DDB the corresponding data block in the compool area is found, and the received asynchronous message can be transferred to the data block.

B.4 SYSTEM INITIALIZATION

Initialization of the DAIS system is performed in three stages: start-up self-testing, configuration identification and software verification/loading. During start-up self-testing, each processor verifies the operational status of all its internal circuits and programs. When a processor completes the start-up procedure it generates a GO signal which is maintained as long as no errors are detected. The second stage, configuration identification, is the procedure which the system undergoes to establish which processors are operational and to select one of 16 possible processor

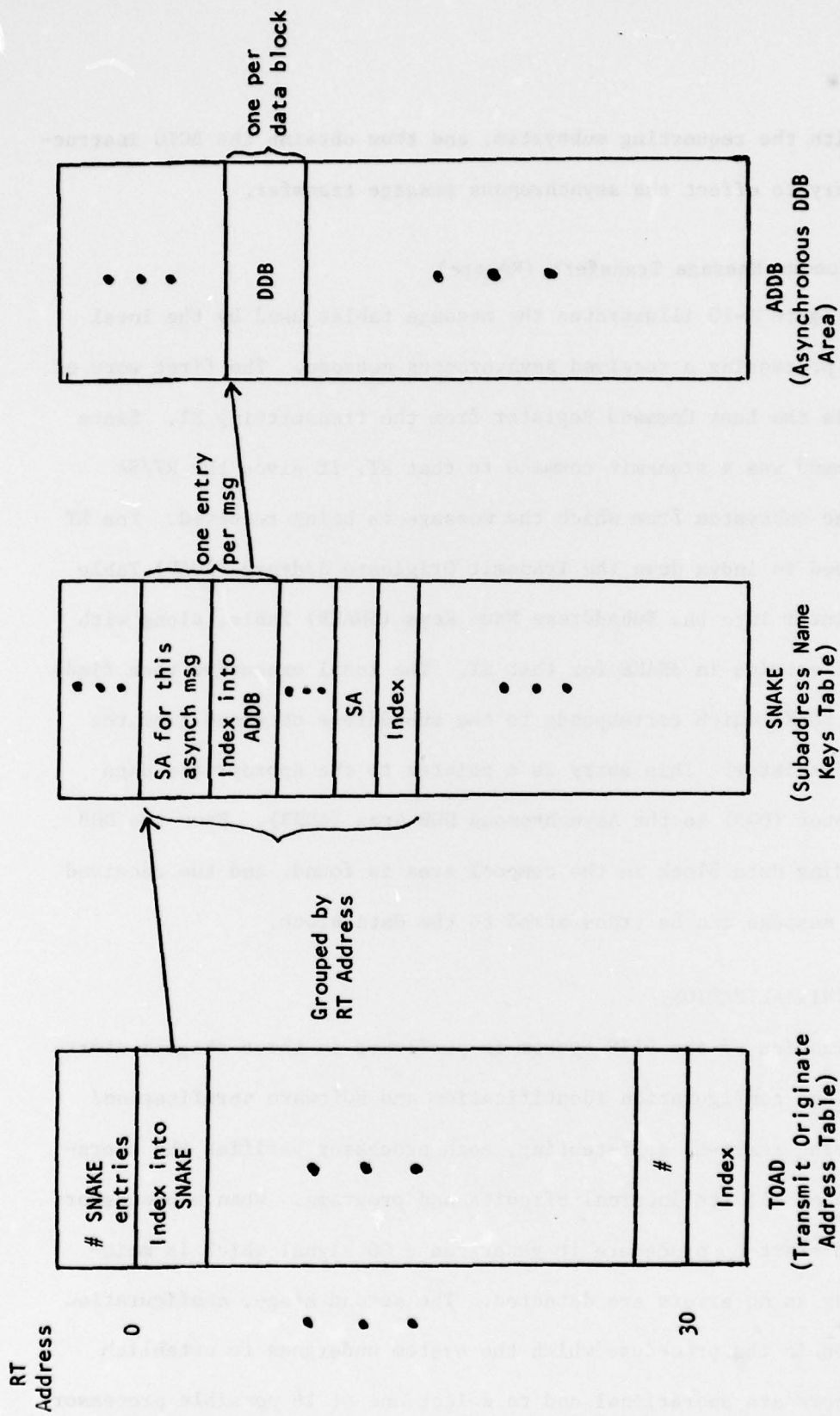


FIGURE B-10. ASYNCHRONOUS MESSAGE TABLES (REMOTE)

configurations. The third phase of the initialization procedure is entered after the configuration index has been generated and the status blocks are obtained from all operational processors. During this phase, each processor retrieves a load module from the system memory and proceeds to verify its validity.

When verification is complete, the master processor initializes the BCIUs of the configured processor, transfers to local executives the start-up location, set its own BCIU to the master mode, and assumes control of the system. The master processor then initiates polling of all remote devices on the multiplex bus. For a remote terminal this procedure consists of sending a sequence of mode commands to first reset the bit register, next to initialize the terminal, and finally to reset the bit register once again. Each command is sent on both buses.

B.5 ERROR RECOVERY

The method of processing communications errors in DAIS usually involves a retry of the original bus commands. However, some errors require intervention by the master executive to send mode commands to the RT involved to investigate the error. The procedure for handling these errors is described in Section 4.3.3 of MA 201 200.

B.6 TCU AND INTERFACE MODULES

Basic timing and control functions of a DAIS remote terminal (RT) are performed by the timing and control unit (TCU). This unit processes information received over the DAIS multiplex bus, prepares proper re-

sponses to this information, and controls the flow of information to the interface modules (IM) of the RT. It is this latter function and the performance of self tests which are of concern to us at the moment, so we shall concentrate on the TCU/IM interface.

Interface modules are signal oriented I/O ports of the TCU. Each IM hardware contains most of the RT circuits which are interface signal dependent. A list of subsystems signals supported by existing DAIS IM cards is presented in Table B-1.

Each IM is implemented as a printed circuit and with a specific signal interface function. DAIS remote terminals support 17 distinct interface modules which are used to input or output signals of the types listed on Table B-1.

The interface modules are designed as dedicated peripherals of the TCU. As such they have no intelligence other than the ability to perform sequential control. The interface between the TCU and the IMs is implemented through an I/O bus which contains the following signals:

- . Clocks
- . IM select code (card slot address)
- . Commands (including channel select and self-test)
- . Data output bus (8 bits + parity)
- . Analog input bus
- . Data input bus (serial)
- . IM identification (type)
- . Handshake (including command acknowledge)
- . Activity and parity

The principal role of the TCU is the routing of data. It does this in response to commands received over the multiplex bus, these commands falling

TABLE B-1 DAIS SUBSYSTEM INTERFACE SIGNALS

Signal	Levels	Input		# Ports/IM	Output	
		Time Between Samples			Max Set Time	# Ports/IM
Discrete	Logic 0					
	Logic 1					
	0V	512 μ sec		32	4 μ sec	32
	-5V	256 μ sec		16	2 μ sec	16
	Open	256 μ sec		16	64 msec nominal	16
Analog	Open	256 μ sec		16	80 msec failure	
	Closed				Not Available	
	-10V to 10V	10 μ sec		8	8 msec refresh	4
	To 11.8V RMS	1.25 msec		8	1.25 msec	8
	11.8V RMS Synchro	1.25 msec		8	1.25 msec	4
Serial	TTL Levels	10-100 msec		4	10-100 msec	4
	TTL levels	Data dependent		4	Data dependent	4
Pulse Counter/Torquer Facility Interface	Special purpose - Not applicable to this report.					

into one of three classes: receive commands, transmit commands, and mode commands. Each command consists of either a subaddress (SA) and a word count or a mode code designator and a mode code number. Thus, a receive command directs the TCU to receive a number (from 1 to 32) of words and to pass these on to one of the IM's according to a mapping specified by the subaddress in the command. The TCU accepts the words from the multiplex bus, stores them until it has been able to verify the accuracy of the transmission, then transmits them to appropriate IM's via the 8-bit output data bus (DOB).

Actions corresponding to a transmit commands are similar but reversed. The interface modules corresponding to the subaddress specified mapping are interrogated and the data from them is received through either the serial digital input bus (DIB) or the analog input bus (AIB). In the latter case, the data is converted to digital form by the analog-to-digital converter within the TCU. In either case, the data is immediately transmitted to the multiplex bus.

The mapping of subaddresses into a corresponding list of IM/channel pairs is implemented with an EROM in the remote terminal. This mapping is established in terms of DAIS required data groupings and the location of interface modules within the RT chassis.

B.7 INTERFACE MODULE TESTING

The RT performs four kinds of self tests: power-on tests, in which the condition of the TCU microcontroller is determined prior to any attempts to service system commands; background self-tests, which test the TCU data

flow elements whenever the RT is in an idle period; data processing tests, which occur during the actual execution of operations in response to a system command, and which determine whether various elements are responding within their allotted time interval; and IM self-tests which test the IM hardware.

For input IM's, self-testing is done via an auxiliary channel, which usually is identical to the main channel, except that in the case of analog channels it has lower accuracy. Output IM's also have an input channel for the purpose of wrapping the output back to the TCU. In either case, the IM self-test is performed automatically after each transmit or receive command from the DAIS system. After the operations directed by a system command have been finished, the TCU tests each IM that was used for that command. Failures detected are stored in the built-in-test (BIT) word for transmission over the multiplex bus upon request.

The actual test performed depends on the type of IM being tested. In the case of IM's for discrete input (differential 5 V DC discrete input, switch closure input, momentary open/ground input, and single-ended 5 V DC discrete input), the word in the data input register is exclusive-ORed with the word in the test input register and the result is transmitted to the TCU via the DIB. Nonzero bits in this word indicate a hardware failure. When the self-test is requested by the TCU, the data input and test input registers are loaded with preassigned words and the result is transmitted to the TCU for verification.

Discrete output IM's (differential and single-ended 5 V DC discrete

output) have wrap-around circuitry in which comparators determine the state of each output line. These states are held in a test register on the IM until a self-test request causes them to be sent back to the TCU over the DIB. The TCU compares the word returned with the output word previously sent to the IM to test for failure.

With the exception of IM's for synchro signals, all analog input IM's (differential DC analog input and differential AC analog input) have a separate, low-accuracy test channel for each data channel. Both data and test signals are multiplexed to the AIB. When the TCU tests an IM channel, the ADC digitizes the signals from the data channel and from the test channel. The digital values are compared by the TCU, and a failure is declared if they differ by more than ten percent of full scale.

The analog output IM's (differential DC analog output, and differential AC analog output) have output signals multiplexed back to the AIB, from which they are digitized by the ADC. The digital values are compared with the requested output values to determine failures.

APPENDIX C

SYSTEM SOFTWARE

CONTENTS

SECTION	PAGE
C. SYSTEM SOFTWARE	184
C.1 DEVICE MANAGEMENT	186
C.2 TASK OPERATION	195

C. SYSTEM SOFTWARE

System software for the Link Module (LM) is shown schematically in Figure C-1. The central feature is the system executive (EXEC), which is responsible for initiating, scheduling, suspending and terminating tasks, and for maintaining certain resources including the system clock. There are three key coroutines in EXEC. The clock of Time Controller (TIM) maintains the clock. The Executive Task Initiator (ETI), is called whenever a task is to be initiated or scheduled. The Executive Task Scheduler (ETS) is responsible for all tasks which are active. When the time at which a task is scheduled for initiation passes, TIM moves it to the READY state, thus making it active.

All I/O operations are managed by the device handlers, of which there are three, one each for the Shared Memory (SM), the Nameplate Interface Controller (NIC), and the Interface Control Adapter (ICA). A dummy handler whose device is a block of memory allows data I/O tasks to use the LM memory as data source or sink.

Each system task is associated with a system task module, as exemplified by the Data I/O Task Module (IOTM), and the NP Initialization Module (NPIM). IOTM is somewhat elaborate since it is reentrant and it relies on subsystem dependent, user-written Data Conversion Programs (DCP). This module is discussed in a later section. Subsystem initialization, error analysis and diagnosis is handled by the user-written Subsystem Diagnostic Module (SDM).

The organization of system software for the Link Manager (LMG) is similar to that for the LM. There are, of course, different device handlers-- one each for the Shared Memory (SM), the Link Control Unit (LCU), and the

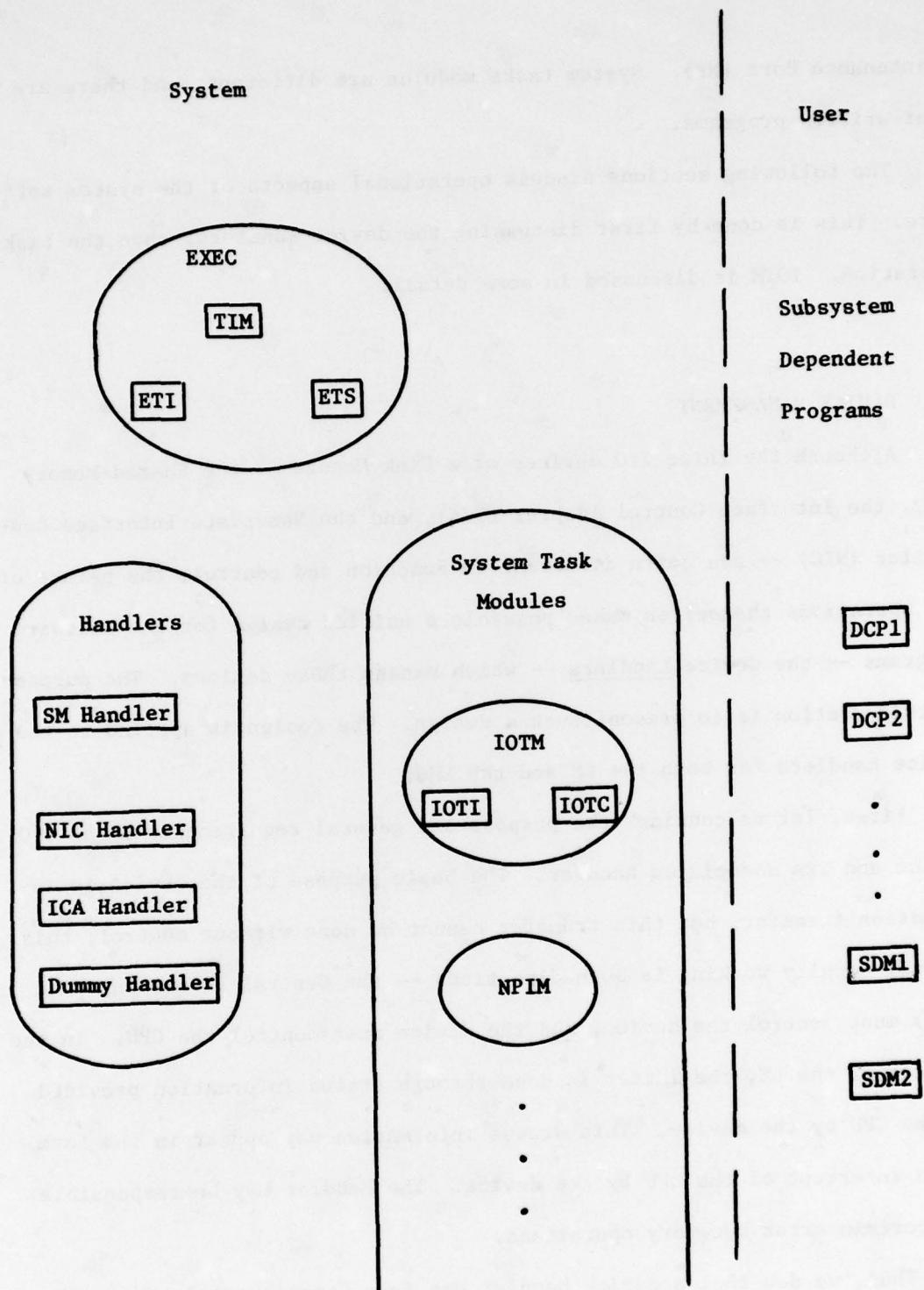


Figure C-1 LM System Software

Maintenance Port (MP). System tasks modules are different, and there are no user-written programs.

The following sections discuss operational aspects of the system software. This is done by first discussing the device handlers, then the task operation. IOTM is discussed in some detail.

C.1 DEVICE MANAGEMENT

Although the three I/O devices of a Link Module -- the Shared Memory (SM), the Interface Control Adapter (ICA), and the Nameplate Interface Controller (NIC) -- are quite different in function and control, the nature of I/O operations themselves makes possible a unified design for the software programs -- the device handlers -- which manage these devices. The purpose of this section is to present such a design. The design is applied to the device handlers for both the LM and the LMG.

First, let us consider the purpose and general requirements of an I/O device and its associated handler. The basic purpose of the device is information transfer, but this transfer cannot be done without control, this control usually working in both directions -- the Central Processing Unit (CPU) must control the device, and the device must control the CPU. In the devices of the LM, the latter is done through status information provided to the CPU by the device. This status information may appear in the form of an interrupt of the CPU by the device. The handler may be responsible for certain error recovery operations.

Thus, we see that a device handler has four functions related to the hardware:

1. device control,
2. transfer of data to/from the device,
3. device status review including interrupt handling,
4. management of certain error recovery operations.

In some cases these functions may overlap.

Usually I/O operations are rather slow compared with CPU operations. The device handler is responsible for matching the speeds, preferably in an efficient manner. Therefore, if a task is waiting for the CPU, a device handler should not retain control of the CPU while long I/O operations are in progress, but it should release it until the I/O operations are finished (as indicated by a device interrupt, for example). More generally, if a device handler is responsible for the I/O operations of several tasks at one time, then the device handler should permit the I/O operations of the tasks, the I/O operations managed by other device handlers, and a task requiring control of the CPU to proceed concurrently. We will now describe the way in which this is done.

By device-process we will mean the set of CPU operations done in performing the I/O operations for a single request by a task. A device-process owns certain resources which are given to it either by the requesting task (task resources) or by the device handler (handler resources). The task resources include a buffer (B#) for the data transferred, a set of instructions given in a control table (CT#), and some memory address space (MAS#). The handler resources include all the hardware assigned to the device-process. Some resources may be shared with other device-processes rather than owned.

A device-process is controlled by the device handler. This program comprises three major parts, the Process Manager (PM), the Traffic Controller (TC), and the Interface Communicator (IC). Each of these is divided into parts: PM into an Initiator (PM.I), a Resource Allocator (PM.R), and a Continuator (PM.C); TC into a Completion Evaluator (TC.C), and an Interrupt Service Routine (TC.I); and IC into an Initiator (IC.I), a Continuator (IC.C), and an Error Processor (IC.E). These eight parts operate as coroutines. A schematic diagram of a device-process and the coroutines of the device handler appears in Figure C-2. Flow of control in the device handler is shown in Figure C-3.

A task requests I/O operations, hence a device-process, by passing the task resources and control to PM.I. At this point the device process becomes active, and it remains active until it is terminated. An active device-process is in one of five states: REQUESTED, WAITING-FOR-RESOURCES, RUNNING, WAITING, and READY.

A new device-process is placed in the REQUESTED state while PM.I performs initialization, then it enters the WAITING-FOR-RESOURCES state until PM.R is able to allocate the necessary handler resources. Then it enters the RUNNING state in which device control and data transfer are done by IC.I and IC.C. If some I/O operations cannot be completed immediately, it enters the WAITING state until TC.C determines that all current I/O operations have been completed, then it enters the READY state and it is added to the Process Ready Queue (PRQ). Whenever a device-process is taken out of the RUNNING state, PM.C moves the next queued device-process (if any) to that state. The cycle -- RUNNING, WAITING, READY, and back to RUNNING -- is

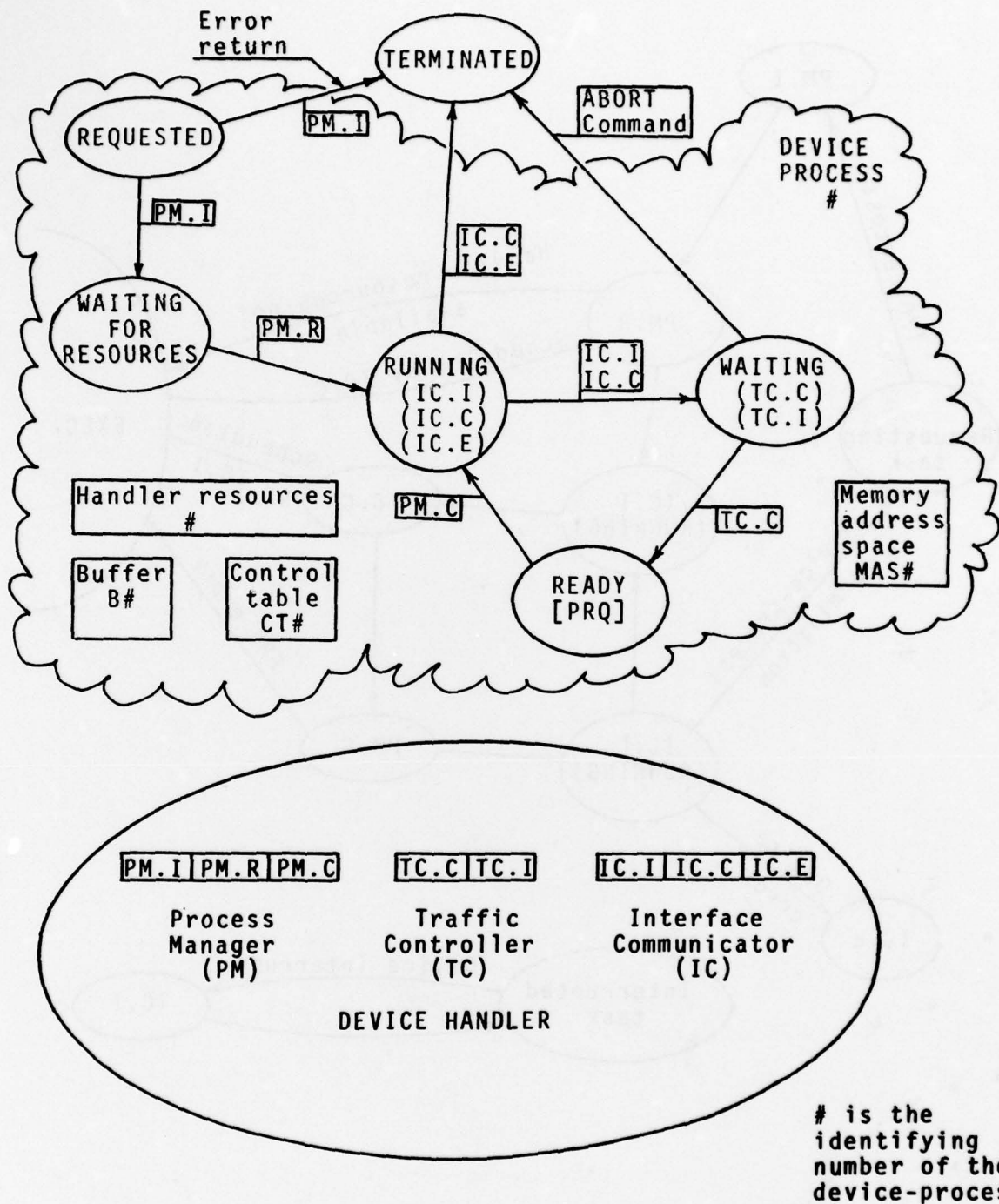


Figure C-2. Device-process and controlling programs.

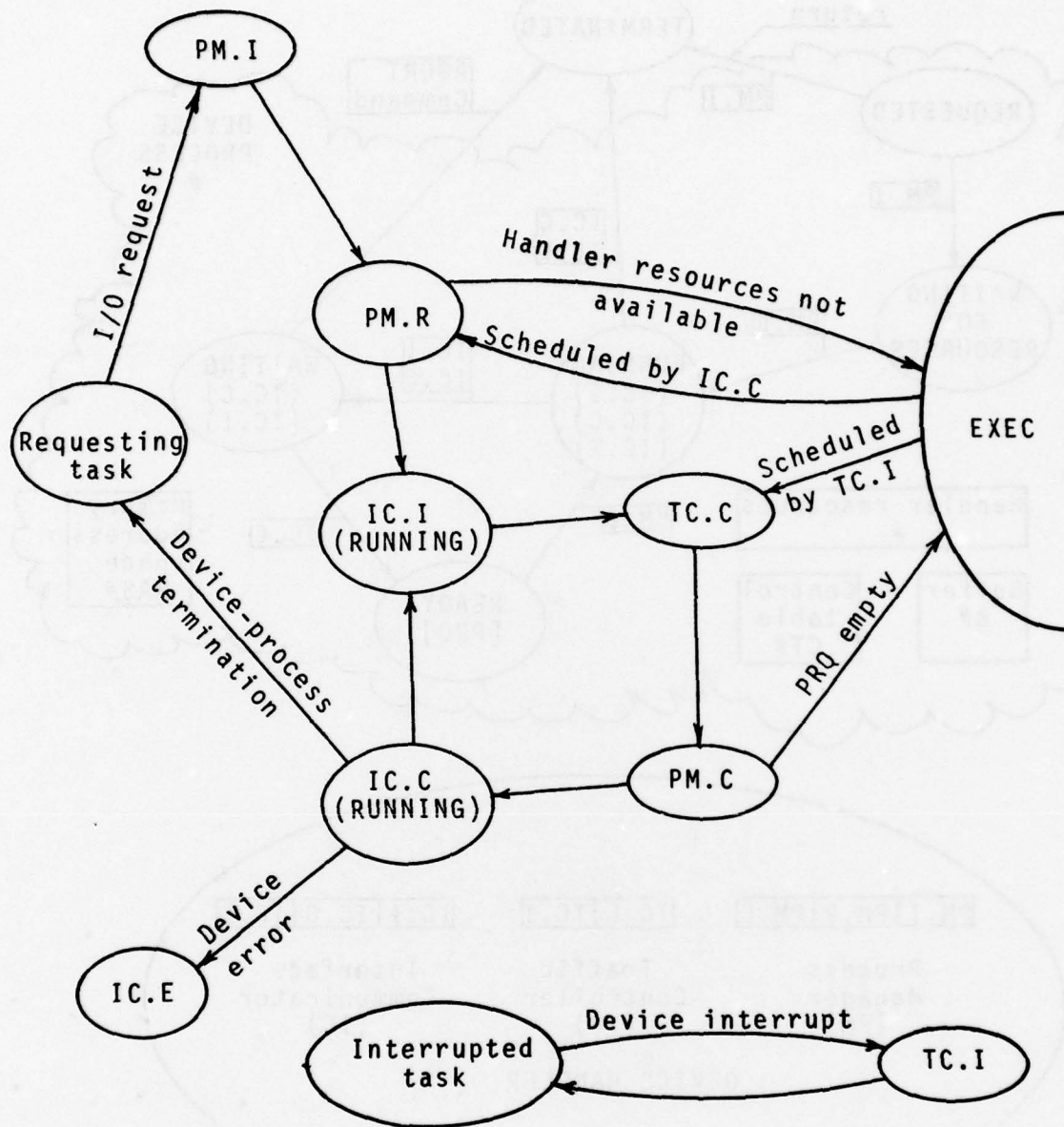


Figure C-3. Flow of control in a device handler.

continued until IC.C determines that the I/O operations for the device-process have been completed and terminates it, returning control and the Hardware Status Word (HWSW#) to the requesting task (or until an ABORT command to EXEC terminates the requesting task). Hardware errors cause control to be passed to IC.E.

Interrupts by the device indicate the completion of certain I/O operations. These interrupts are intercepted and interpreted by TC.I, which records the new status of I/O operations and schedules TC.C. Thus, the device-processes are driven by interrupts and no time is lost waiting for the completion of I/O operations.

The functions of the various coroutines of a device handler are given in Table C-1. Note that hardware errors are evaluated by IC.C and reported to the requesting task via the Hardware Status Word (HWSW#).

C.1.1.1 The SM Handler for the LM

The SM handler for the LM is much simpler than the general handler described in the previous section. Since data transfer is through the SM hardware buffers which are part of the task resources, there is no waiting for completion of I/O operations. Hence the WAITING and READY states are combined into the RUNNING state, and the coroutines TC.C and PM.C become null routines so that control is passed directly from IC.I to IC.C. No detectable device errors are possible, so IC.E is deleted. Coroutine TC.I is responsible for invoking tasks for the LM function commands and the data transfer commands from the LMG to the LM. Invocation of any task transferring data to/from the SM involves updating the buffer address for the task. Since there may be a delay before the buffer becomes available, this is done by a

TABLE C-1
FUNCTIONS OF DEVICE HANDLER COROUTINES

Process Manager

- PM.I - 1. Receives control and task resources from the requesting task.
2. Checks the request for errors and returns to the requesting task with an error report if there are any.
3. Suspends the requesting task (so EXEC can pass control to other tasks when time is available).
4. Resets the CT# pointer.
5. Activates the new device-process and places it in the WAITING-FOR-RESOURCES state.
6. Passes control to PM.R.
- PM.R - 1. Allocates handler resources to the next device-process in the WAITING-FOR-RESOURCES state, if possible. Otherwise, it returns control to EXEC.
2. Moves this device-process to the RUNNING state.
3. Passes control to IC.I.
- PM.C - 1. Places into the RUNNING state the next device-process in PRQ. If PRQ is empty, it returns control to EXEC.
2. Passes control to IC.C.

Interface Communicator

- IC.I - 1. Begins the next (marked by the CT# pointer) I/O operations specified in CT#. Advances the CT# pointer.
2. Moves the device-process to the WAITING state.
3. Passes control to TC.C.
- IC.C - 1. Checks for hardware errors, passes control to IC.E if there are any.
2. Transfers data, if necessary, from the device interface to the buffer (B#).
3. Determines if the I/O operations for this device-process are complete. If so, it terminates the device process, evaluates the hardware status, generates the hardware status word (HWSW#), brings the requesting task out of suspension, and returns control to that task.
4. If the I/O operations for this device-process have not been completed, it passes control to IC.I to start the next I/O operations.
- IC.E - 1. Device dependent error recovery operations.

Traffic Controller

- TC.I - 1. Intercepts device interrupts and determines which I/O operation has been completed.
2. Records the new status of the I/O operation.
3. Schedules TC.C to start after termination of the interrupted task, or schedules special tasks as required by the device.
4. Returns control to the interrupted task.

- TC.C - 1. Determines which, if any, device-processes in the WAITING state have had their current I/O operations completed. Moves these processes to the READY state and adds them to PRQ.
2. Passes control to PM.C.

status request call to the SM handler from the task, the address of the buffer being returned when it becomes available.

C.1.2 The NIC Handler for the LM

The NIC handler can have the structure described for the general device handler, but some simplifications are possible since at most one NIC-using task need be active at any particular time. In this case, the handler resources are always available to a new NIC-process, and the READY state is unnecessary. PM.R can be combined into PM.I and PRQ eliminated, thus simplifying TC.C and PM.C. The error processing coroutine IC.E should initiate at least one retry in the event of parity error.

C.1.3 The ICA Handler for the LM

The ICA requires essentially the full capabilities of the general device handler. This is because several tasks may have ICA-processes running concurrently with a complex mix of ICA channels. Some of these channels (analog output) may be owned by an ICA-process and require no wait for the I/O operation, some (analog input) may be shared and require no wait, some (serial I/O) may be owned and require a wait, and some (momentary contact closure) may initiate a task through an interrupt. The latter can also be handled by a task whose ICA-process waits for the event (contact closure), thus simplifying the scheduling problems of TC.I.

The ICA handler has heavy responsibilities with regard to checking and error detection: all outputs should be checked by wrapping them around through an input channel. For the serial I/O channels this is sufficiently time consuming that it should be possible for the requesting task to override this checking facility.

C.1.4 The MP Handler for the LMG

The LMG Maintenance Port (MP) is a serial I/O port which is managed in much the same way as is the NIC port of the LM. Thus, at most one MP-using task need be active at any particular time, so handler resources are always available to a new MP-process. This leads to simplifications of the type described in Section C.1.2.

C.1.5 The SM Handler for the LMG

The SM handler for the LMG is similar to that for the LM in that there is no waiting for completion of I/O operations. Thus, this handler can be similar in many respects to the SM handler of the LM. Asynchronous message requests from the LM are handled in a manner similar to that of commands from the LMG to the LM. The handshake between two SM commands, one from the LM, one from the LMG, is done through SM command and status words.

C.1.6 The LCU Handler of the LMG

The LCU Handler of the LMG is strictly an interrupt service routine. The handler is responsible for properly vectoring interrupts to various task routines.

C.2 TASK OPERATION

A task is a set of CPU operations which accomplishes something. These operations are defined by and are under control of the task's program or module. For our purposes, a task is initiated by transfer of control from EXEC to the module's initiation entry point and it is terminated by transfer of control from a completion exit point of the module to a completion return

point of EXEC. These transfers will be called the initiation call and the completion return, respectively. Once initiated, a task is said to be active until it is terminated. The CPU is not, however, always performing operations for a task while the task is active. Indeed, a task may be put into a dormant state while certain operations (e.g., hardware I/O operations) are completed. While a task is in such a state of dormancy, the CPU might perform operations for a different task. A task not in a dormant state is in the RUNNING state. Now the vague definition given above can be made somewhat more precise: A task is the set of operations done between the initiation call by EXEC and the termination return by the task's module while the task is in the RUNNING state.

Two tasks are said to be concurrent if they are active at the same time. No two tasks can be in the RUNNING state at the same time, but two tasks can be active concurrently and controlled by the same module. Such concurrency occurs with Data I/O tasks, for example.

There are three possible states for a task - the RUNNING state, the SUSPENDED state, and the READY state. These appear in Figure C-4, a schematic diagram of a task. Figure C-5 shows the flow of control. The task is first put into the READY state and added to the Task Ready Queue (TRQ). Whenever a task is removed from the RUNNING state, the Executive Task Scheduler (ETS) moves the next-queued task in TRQ into the RUNNING state. Once a task is in the RUNNING state, it remains there until an I/O operation is necessary or an I/O interrupt occurs. As soon as the task calls an I/O device-handler, the task is put into the SUSPENDED state, and it remains there until the device handler completes its action, and causes the task to

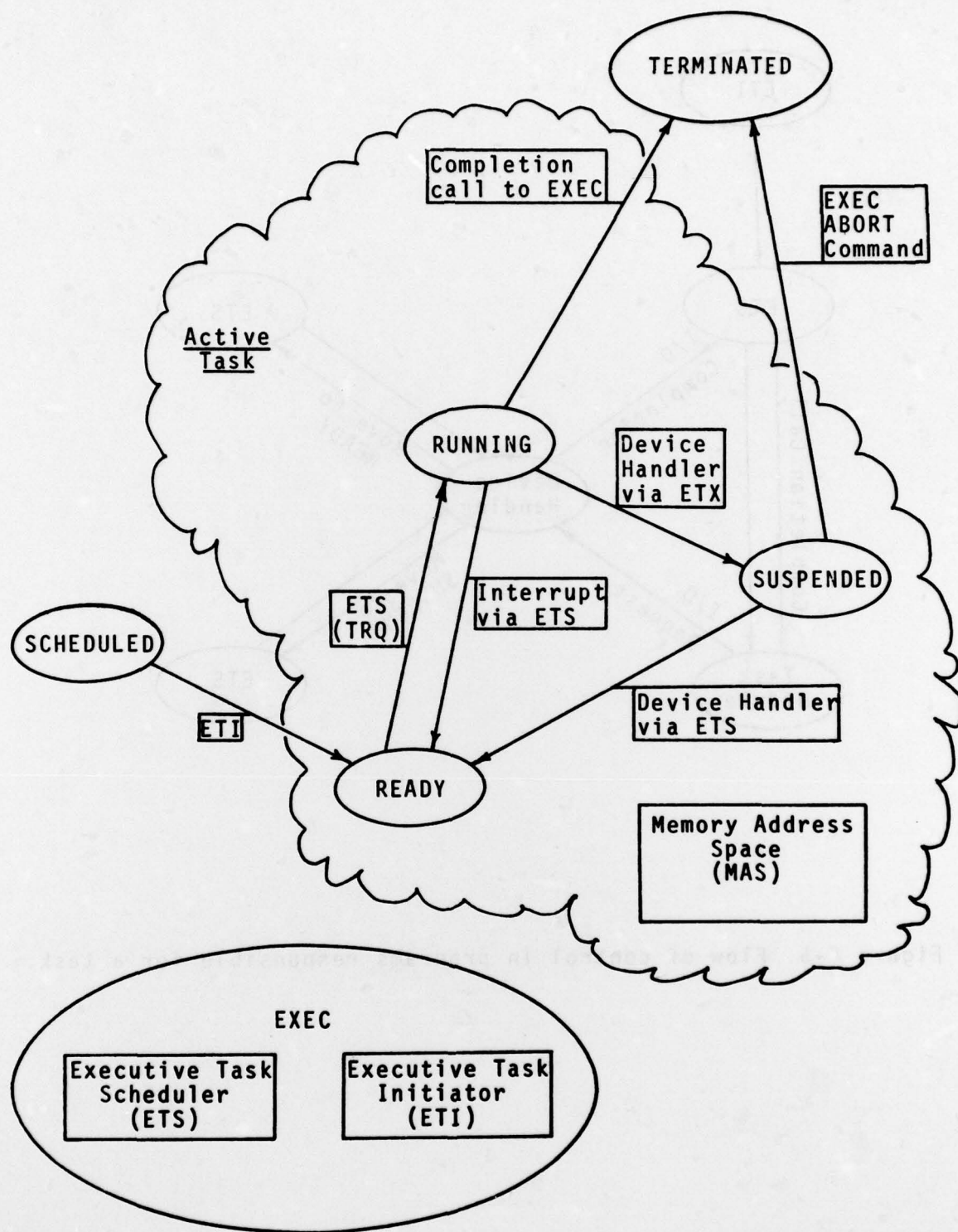


Figure C-4. Active Task and Controlling Programs

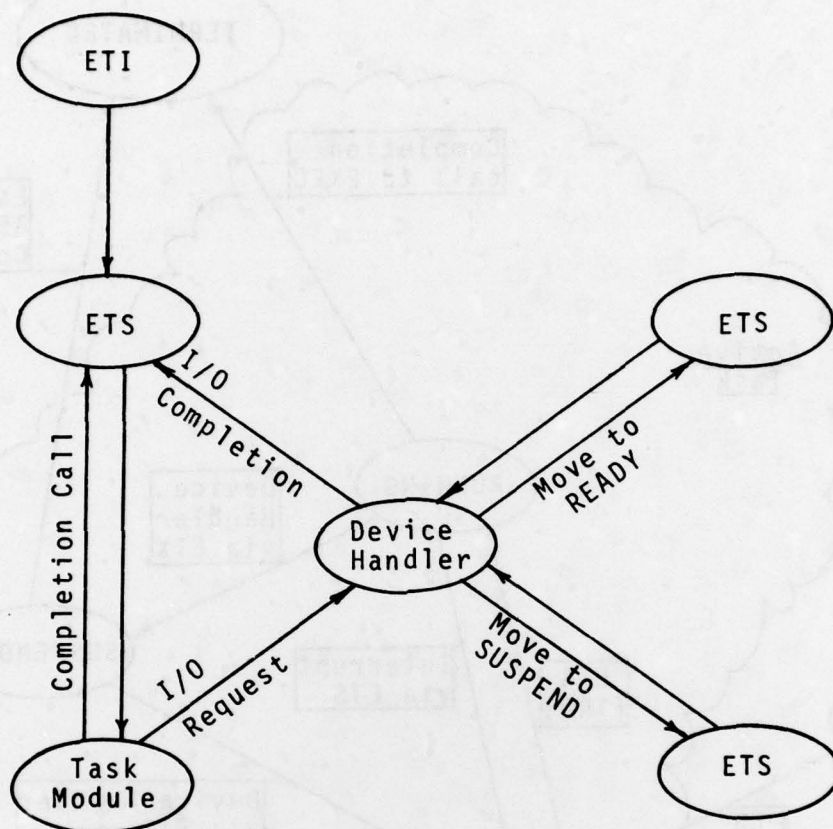


Figure C-5. Flow of control in programs responsible for a task.

be moved to the READY state and added to TRQ. Whenever an interrupt occurs, the task which is in the RUNNING state is moved to the READY state and added to TRQ. These changes of state are all handled by ETS.

It should be noted that each task owns certain resources, including a certain amount of Memory Address Space (MAS). Part of this MAS is used by ETS when it suspends the task.

EXEC provides the following services for handling task operations:

1. Task initiation,
2. Task suspension,
3. Bring task from suspension,
4. Task termination.

Task initiation can be requested in either of two ways:

- a. by command from a high level system (LMG for LM, Dais for LMG), or
- b. by request from an active task.

The latter should permit immediate initiation or scheduled initiation (i.e., at a certain time), and a task should be able to schedule itself (thus becoming repetitive). Finally, provision should be made for

5. Task abortion by I/O timeout or by command from a higher level system.

C.2.1 Data I/O Task Functioning

Data transfer, the primary function of a LM, is handled by the data I/O tasks, each of which should be thought of as a pipeline from a data source to

a data sink. Either the data source or the data sink must be one of the LM data ports (ICA or SM), the other one being the other data port, a block of memory, or empty (see Figure C-6).

A data I/O task operates in the same manner as any other task. The task module comprises two coroutines, the I/O Task Initiator (IOTI) and the I/O Task Controller (IOTC). These will be referred to collectively as the I/O Task Module (IOTM).

IOTI is called whenever an I/O task is to be invoked. The task is identified by its Link Address (LA), which IOTI uses to select the appropriate entry from the Subsystem Configuration Table (SCT). This entry, which drives the I/O task via the IOTC, indicates the resources to be used by this I/O task: the data source, the data sink, the Data Conversion Program (DCP), the Subsystem Diagnosis Module (SDM), memory address space, and the I/O protocols as given in the Input Control Table (ICT) and the Output Control Table (OCT). The memory address space includes two buffer areas, the Task Input Buffer (TIB) and the Task Output Buffer (TOB), either of which can be a buffer area of the SM. IOTI initiates the I/O task, determines (via SM status request) the address of any SM buffer to be used, then transfers control to IOTC, which is responsible for operation of the task.

The flow of data in a Data I/O Task is shown in Figure C-7 and a flow chart for the task appears in Figure C-8. IOTC first requests the data source handler to fill TIB. When this has been done, IOTC calls DCP to convert and transfer the data to TOB. DCP is a task-dependent program. It receives the data source Hardware Status Word (HWSW), so it can evaluate the condition of the data it receives and request remedial action (e.g.,

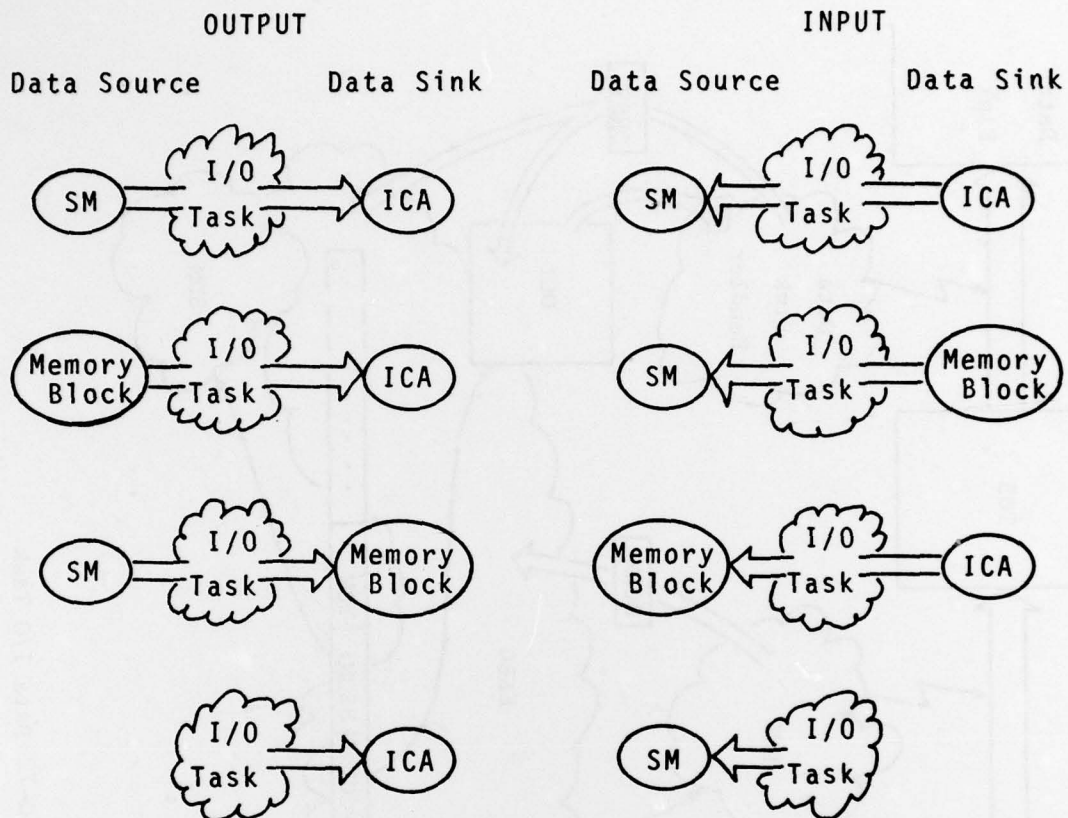


Figure C-6. Data sources and sinks for Data I/O Tasks.

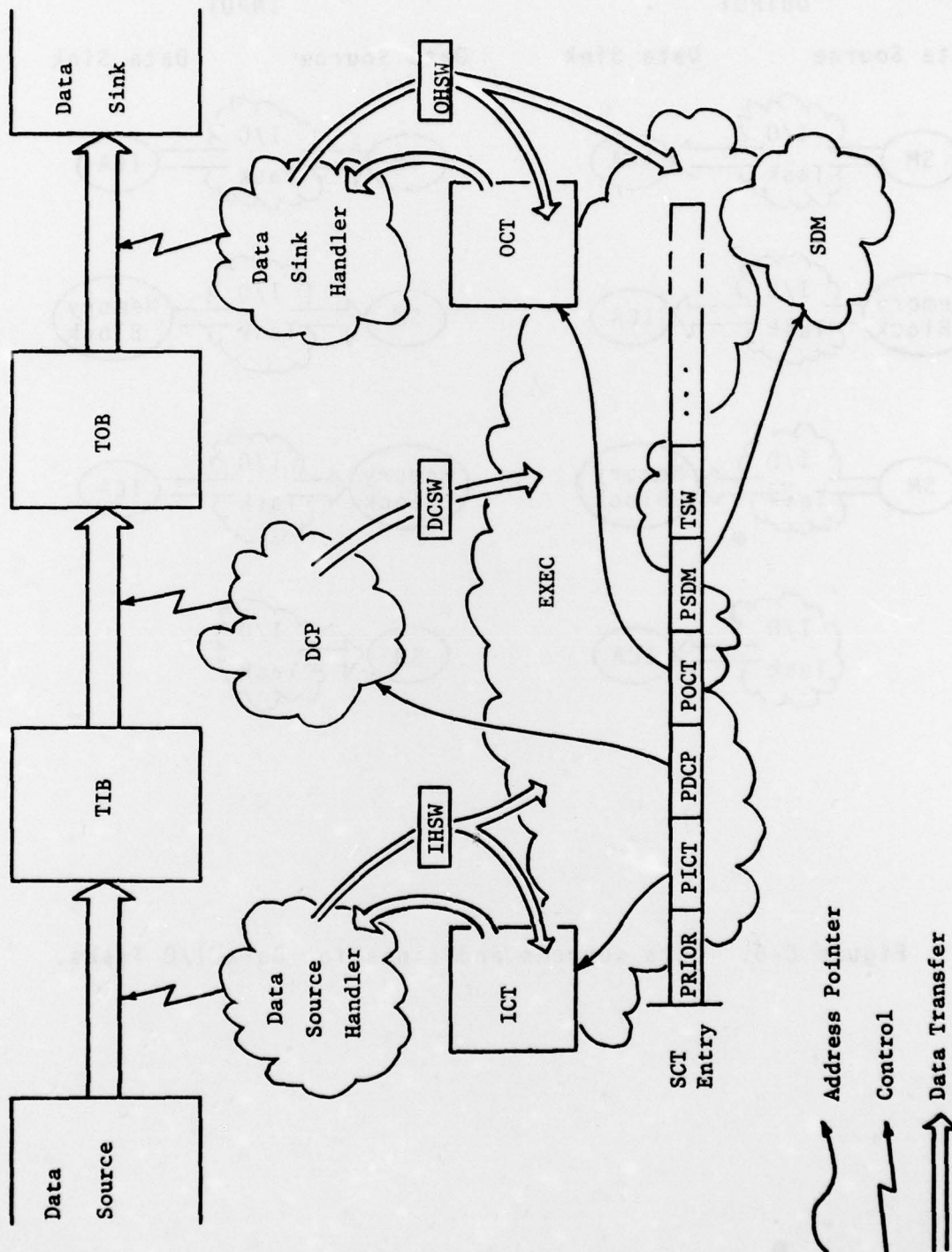


Figure C-7 Data I/O Task

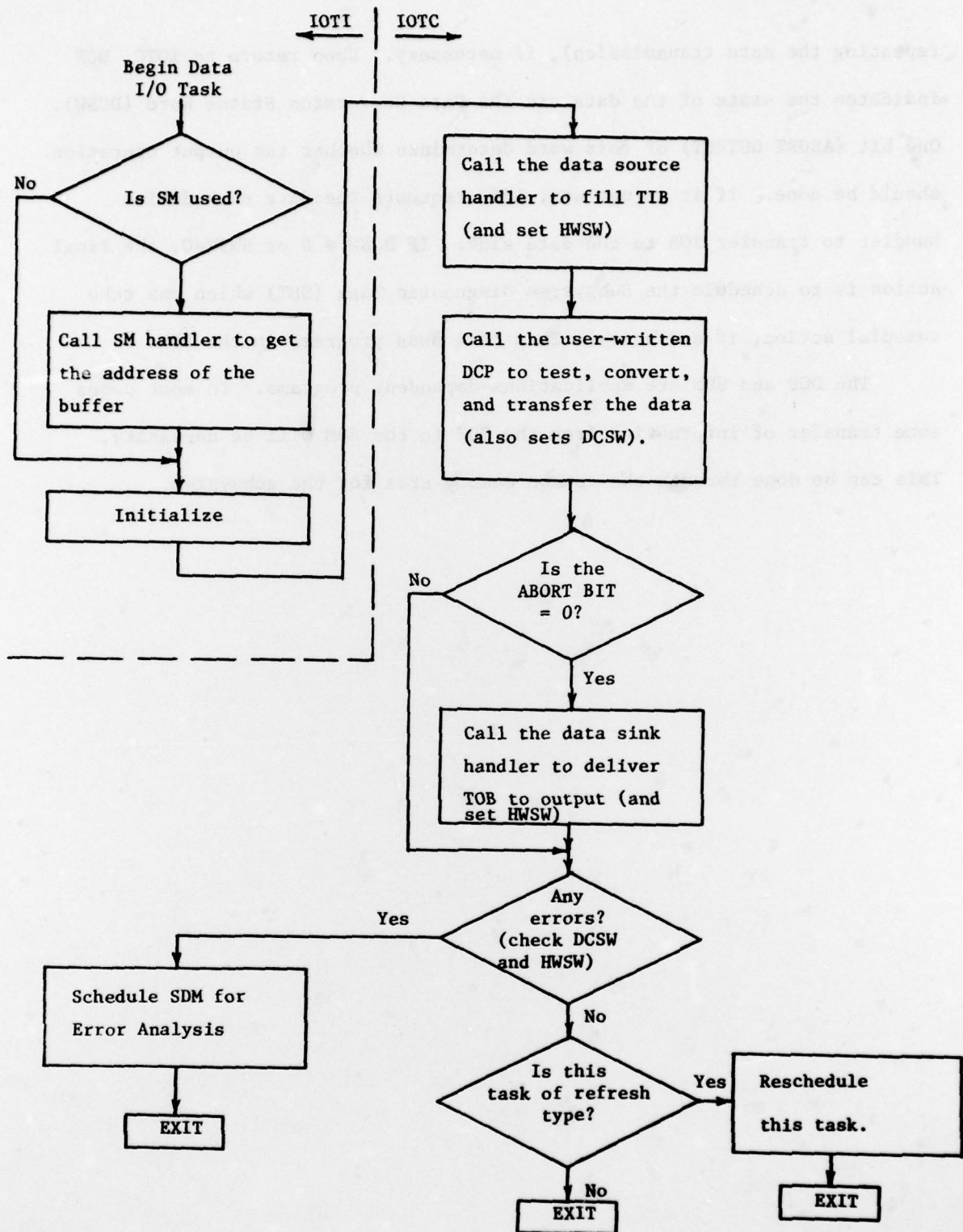


Figure C-8 Data I/O Task Operation

repeating the data transmission), if necessary. Upon return to IOTC, DCP indicates the state of the data via the Data Conversion Status Word (DCSW). One bit (ABORT OUTPUT) of this word determines whether the output operation should be done. If it is not set, IOTC requests the data sink device handler to transfer TOB to the data sink. If DCSW \neq 0 or HWSW \neq 0, the final action is to schedule the Subsystem Diagnostic Task (SDT) which can take remedial action, if necessary. This task uses programs in the SDM.

The DCP and SDM are applications-dependent programs. In most cases some transfer of information from the DCP to the SDM will be necessary. This can be done through the common memory area for the subsystem.

APPENDIX D

PROCESSING SYNCHRO SIGNALS

Appendix D. Processing Synchro Signals

A synchro resolver produces three AC signals

$$v_1 = V_1 \cos \omega t,$$

$$v_2 = V_2 \cos \omega t,$$

$$v_3 = V_3 \cos \omega t,$$

at the frequency and phase of the AC reference signal. Thus, if the signals are synchronously sampled at the time of reference signal peak value and digitized (e.g., in a single group of an ICA, the group being configured for AC A/D conversion), the digital values obtained will be the three amplitudes V_1 , V_2 , and V_3 .

The resolver shaft angle θ is encoded into the signal amplitudes according to the following equations:

$$D.1 \begin{cases} V_1 = A \cos (\theta - 120^\circ), \\ V_2 = A \cos \theta, \\ V_3 = A \cos (\theta + 120^\circ), \end{cases}$$

where A is a constant depending on the resolver and the amplitude of the reference signal. We now derive formulas for conversion of (V_1, V_2, V_3) to the pair $(\cos \theta, \sin \theta)$.

$$\text{Since } \cos 120^\circ = -\frac{1}{2} \text{ and } \sin 120^\circ = \frac{\sqrt{3}}{2},$$

we have

$$D.2 \begin{cases} \cos(\theta-120^\circ) = -\frac{1}{2} \cos \theta + \frac{\sqrt{3}}{2} \sin \theta, \\ \cos(\theta+120^\circ) = -\frac{1}{2} \cos \theta - \frac{\sqrt{3}}{2} \sin \theta, \end{cases}$$

whence,

$$V_1 - V_3 = \sqrt{3} A \sin \theta,$$

and

$$V_1^2 + V_2^2 + V_3^2 = \frac{3}{2} A^2$$

Thus, the equations of the conversion are

$$D.3 \left\{ \begin{array}{l} A = \sqrt{\frac{2}{3} (V_1^2 + V_2^2 + V_3^2)}, \\ \cos \theta = V_2/A, \\ \sin \theta = (V_1 - V_3)/A. \end{array} \right.$$

Resolver data can be checked by testing the quantity A and/or the sum $V_1 + V_2 + V_3$. A zero value of A indicates no resolver signal: a value of A outside the range specified for the resolver may indicate resolver failure. The sum $V_1 + V_2 + V_3$ should (in theory) be zero, so significant deviation from zero indicates a failure. A test based on $V_1 + V_2 + V_3 = 0$ should guard against the condition $V_1 = V_2 = V_3 = 0$ (e.g., by testing A).

If θ is needed, an arc tangent computation can be used to determine it from $(\cos \theta, \sin \theta)$. For most applications this is not necessary.

Synchro signals are generated by an ICA group configured for AC D/A conversion if the digital values loaded into the three channels are V_1 , V_2 , and V_3 given in equations D.1. All these can be evaluated in terms of $\cos \theta$ and $\sin \theta$ (see D.2).